√ 06.12 - 11/01/96 - 02352

Operation and Maintenance Manual MCB Camp Lejeune Groundwater Treatment System

Volume VI of VII

Submitted to:

DEPARTMENT OF THE NAVY
Contract No. N62470-93-D-3032

Submitted by:

OHM Remediation Services Corp.
A Subsidiary of OHM Corporation

5335 Triangle Parkway, Suite 450
Norcross, GA 30092

OHM Project No. 16032

November 1996

✓

06.12-11/01/96-02352

TABLE OF CONTENTS

Volume I

TABLE OF CONTENTS - CONTINUED

TABLE OF CONTENTS - CONTINUED

TABLE OF CONTENTS - CONTINUED

TABLE OF CONTENTS - CONTINUED

APPENDIX E

VOLUME VI

LIST OF CONTRACTOR/SUBCONTRACTORS/MANUFACTURERS/SUPPLIERS

OHM Remediation Services Corporation
5335 Triangle Parkway, Suite 450
Norcross, Georgia 30092
(404) 729-3900

Southerland Electric Company
2815 Commerce Road, P. O. Box 626
Jacksonville, N.C. 28541-0626
(919) 347-1754

GE Fanuc Automation North America, Inc.
P. O. Box 8106
Charlottesville, Virginia 22906
1-800-828-5747

Honeywell
Industrial Automation and Control
Automation College
2820 West Kelton Lane
Phoneix, Arizona 85023
(602) 789-5669

Rosemount Analytical Inc.
2400 Barranca Parkway
Irvine, California 92714
(714) 863-1181

Drexelbrook
205 Keith Valley Road
Horsham, Pennsylvania 19044
(215) 674-1234

Signet Scientific Company
3401 Aerojet Avenue
El Monte, California 91731-2882
(818) 571-2770

W. E. Anderson Division
Dwyer Instruments, Inc.
P. O. Box 358
Michigan City, Indiana 46360
(219) 879-8000

Process Control Services, Ltd.
2200 Seaford Road, P. O. Box 98
Seaford, Virginia 23696-0098
(804) 898-4332

Soil and Groundwater Remediation Plant

Operable Unit No. 2
MCB Camp Lejeune, North Carolina

Operation and Maintenance Manual

Programmable Logic Controller System

Table of Contents

Safety Precautions

Danger

Control panel contains voltages that are dangerous to life. **Do Not** touch exposed conductors or components without verifying the circuit is de-energized.

Warning

Protect circuit boards from damaging transients by removing all power to circuit before removing or inserting circuit boards.

Warning

Electronic components are extremely static sensitive. Damage to circuit boards can occur by static discharge when handling. Observe proper personnel grounding if electronic boards must be handled.

Environmental Conditions

Component life is greatly affected by temperature and humidity conditions within the control enclosure. Although components within the control enclosure are rated for 50 C (121 deg. F.) and 90% humidity (non-condensing), these conditions are only for short duration. Significantly better performance can be achieved by maintaining temperature within the enclosure below 38 C (100 deg. F.) and 70% humidity.

Low temperature combined with high humidity can cause condensation of water vapor on circuit boards and entrance conduits, leading to irreversible damage to circuit components. This condition can be avoided by maintaining the temperature in the control room above the dew point. In climates where the humidity routinely is above 80%, it is recommended that humidity and temperature meters be mounted in the vicinity of the control enclosure as an operator guide for operation of the control room's climate control system.

Preventive Maintenance and Schedule

Batteries - The PLC central processing unit (slot 1) and the Ethernet communications unit (slot 2) each have a lithium battery to maintain their programming over extended shutdown periods. These batteries should be replaced every 5-10 years.

Troubleshooting Guide and Diagnostic Techniques

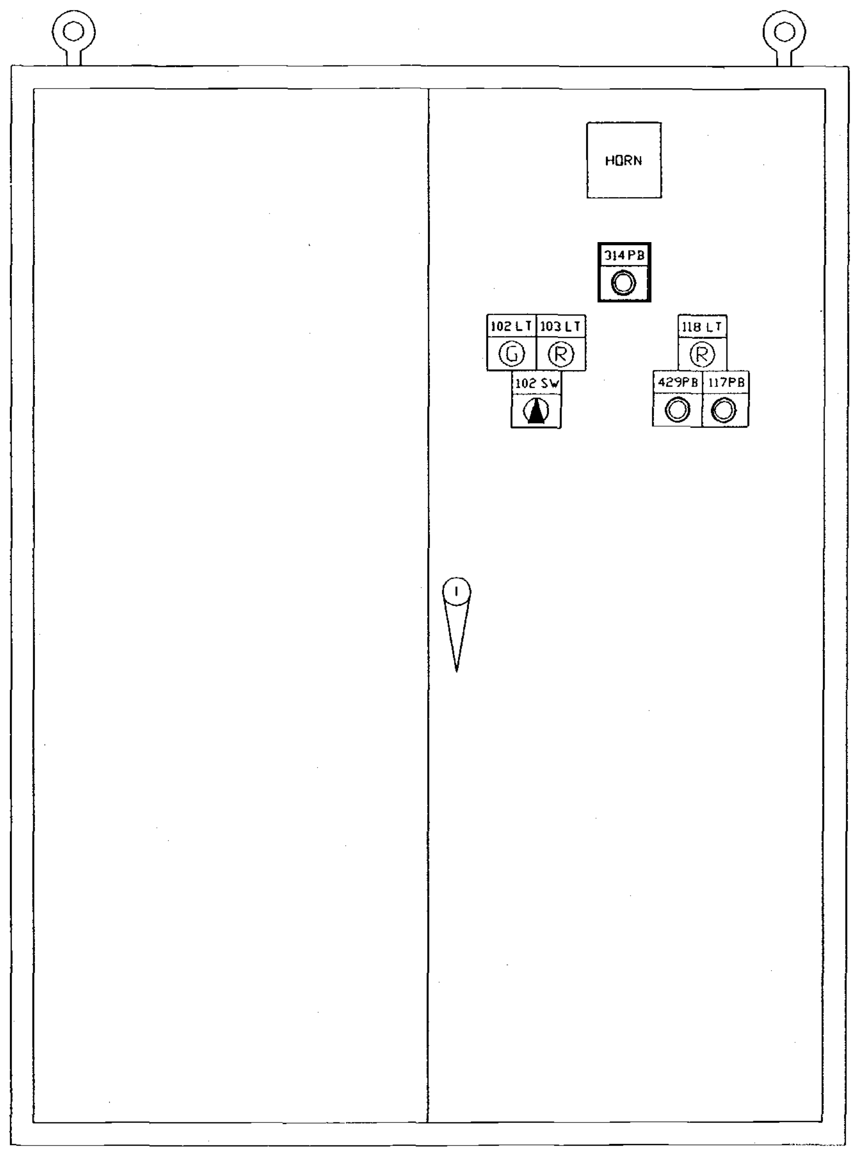
Symptom/Problem	Action
PLC running light not illuminated	• Check Power. • Check PLC CPU status lights. If not all illuminated, verify that CPU mode switch is in RUN-OUTPUTS ENABLED. Reset PLC by turning off power for 10 seconds.
Ethernet card status lights not all illuminated or PC workstation not communicating with PLC.	• Press module reset push button behind door of Ethernet card to reinitialize. • Verify Ethernet cable is connected and AIU-10BASE2 converter is operating. Ensure that converter selector switch is SQE ON.
Input and/or output module status lights flashing.	• Reset PLC by turning off power for 10 seconds.
Analog input signal reads zero.	• Check 24 VDC power supply fuse. • Check wiring.
Analog output does not drive load.	• Check 24 VDC power supply fuse. • Check loop power fuse.
Digital output module FUSE light is illuminated.	• Check output fuse inside module. • Correct short circuit condition.
PLC not communicating with wells.	• Check 5 VDC power supply fuse. • Check 24 VDC power supply fuse. • Check telemetry wiring for proper connection. • Check well power.
Motor does not run when controlled from PC workstation.	• Verify that motor has power, overloads are reset, and H-O-A switch is in AUTO. • Check that motor will run in HAND. Motor starter control power fuse might be blown. • Check that PLC is energized. • Check that PC workstation is communicating with PLC.
Telephone dialer problems.	• Verify that PLC is running. • Check wiring between PLC and dialer. • Consult dialer manual and/or manufacturer.

If the above troubleshooting chart does not solve a problem with the PLC, consult the *Series 90-70 Programmable Controller Reference Manual* for more detailed troubleshooting of the hardware. If using a laptop computer running *Logicmaster 90-70* software, online help will be available. Consult the *Logic-master 90-70 Programming Software Users Manual* for detailed instructions on diagnostic procedures.

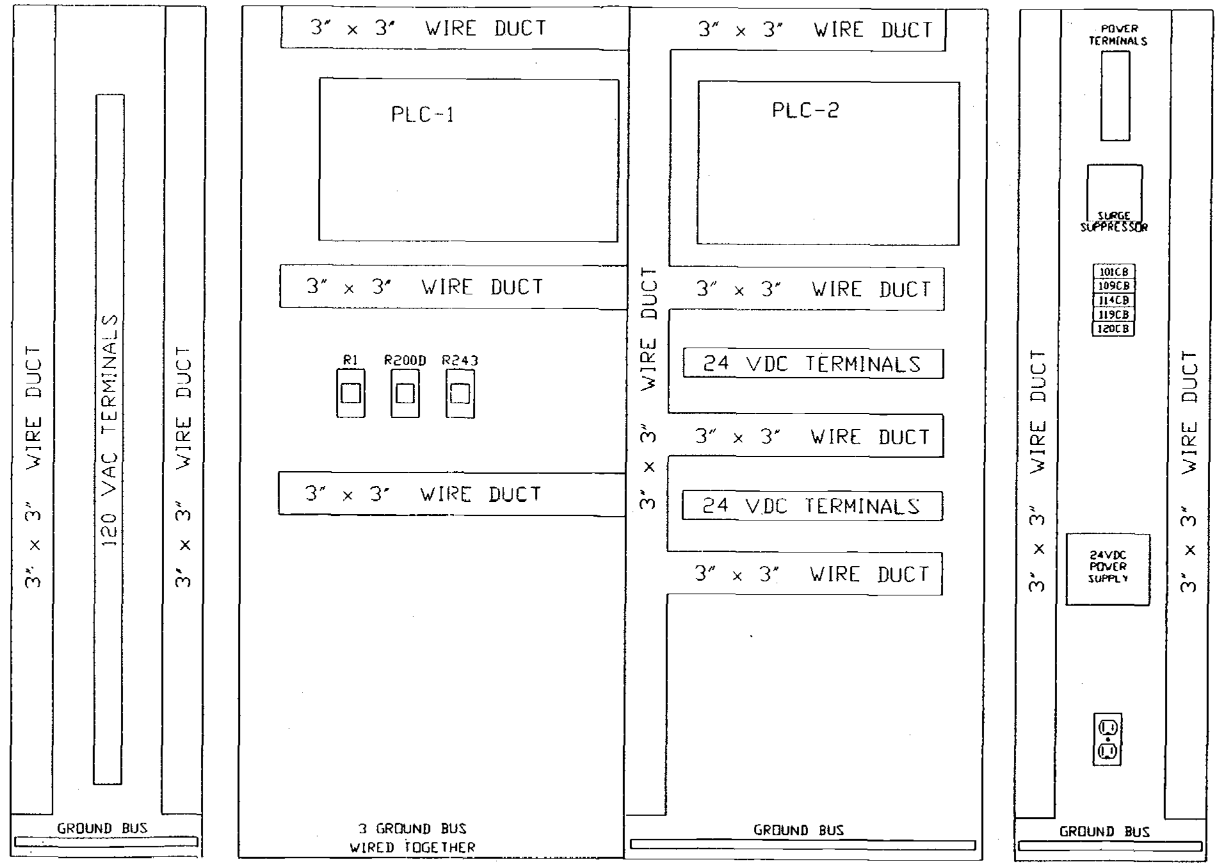
There is a complex series of interlocks programmed to prevent equipment damage and contamination. The operator should become familiar with these interlocks to preclude mistaking an active interlock for an equipment failure.

Interlock Table

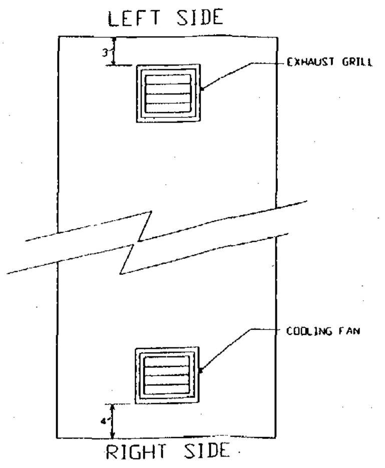
Interlock Description	Interlock Function
I-110 Low water level T-110	• Stop air stripper feed pumps P-110A/B • Stop jet mixing pump P-120 • Stop sulfuric acid pump P-211
I-121 High water pH in X-130	• Stop caustic feed pump P-121
I-122 High caustic level in tank T-121	• Send off alarm
I-123 Low caustic level in tank T-121	• Stop caustic feed pump P-121 • Stop all shallow well pumps • Stop polymer feed pump X-132 • Stop spent backwash pump P-205
I-145 Low water in tank T-145	• Stop supernatant transfer pump P-145
I-146 High water level in head tank T-145	• Stop caustic feed pump P-121 • Stop all shallow well pumps • Stop polymer feed pump X-132 • Stop spent backwash pump P-205
I-150 High-high water level in tank T-110	• Stop sulfuric acid pump P-211 • Stop all deep well pumps
I-151 High water level in tank T-110	• Stop supernatant transfer pump P-145
	• Stop all shallow well pumps • Stop polymer feed pump X-132 • Stop backwash water pump P-205
I-152 Air pressure below 60 psig.	• Stop GAC feed pumps P-220A/B
I-200 Air stripper blower K-200 low pressure	• Stop air stripper feed pumps P-110A/B
I-201 Low (below 6.5) pH in tank T-110	• Stop sulfuric acid feed pump P-211
I-205 High level backwash holding tank T-205	• Stop backwash pump P-241
I-206 Low level backwash holding tank T-205	• Stop spent backwash pump P-205
I-211 Air stripper feed pump P-110A/B shutdown	• Stop sulfuric acid feed pump P-211
I-212 Low level sulfuric acid tank T-211	• Stop sulfuric acid feed pump P-211 • Stop all deep well pumps • Stop supernatant feed pump P-145
I-213 High level sulfuric acid tank T-211	• Send off alarm
I-220 Low level stripper tank T-220	• Stop GAC feed pumps P-220A/B • Stop blower K-200 after 15 min. time delay
I-221 High level stripper tank T-220	• Stop air stripper feed pumps P-110A/B • Stop blower K-200 after 5 min. time delay
I-222 High-high level stripper tank T-220	• Stop blower K-200 immediately
I-240 Low level effluent tank T-240	After 5 min. delay • Stop backwash pump P-241 • Stop reuse/seal water pump P-245 • Stop GAC feed pumps P-220A/B • Stop air stripper Feed pumps P-110A/B • Stop supernatant feed pump P-145 • Stop spent backwash pump P-205
I-242 High level effluent tank T-240	• Stop GAC feed pumps P-220A/B
I-245 Reuse/Seal water pump P-245 shutdown	• Stop backwash pump P-241 • Stop reuse/seal water pump P-245 • Stop GAC feed pumps P-220A/B • Stop air stripper Feed pumps P-110A/B • Stop supernatant feed pump P-145 • Stop spent backwash pump P-205



FRONT VIEW



INTERIOR



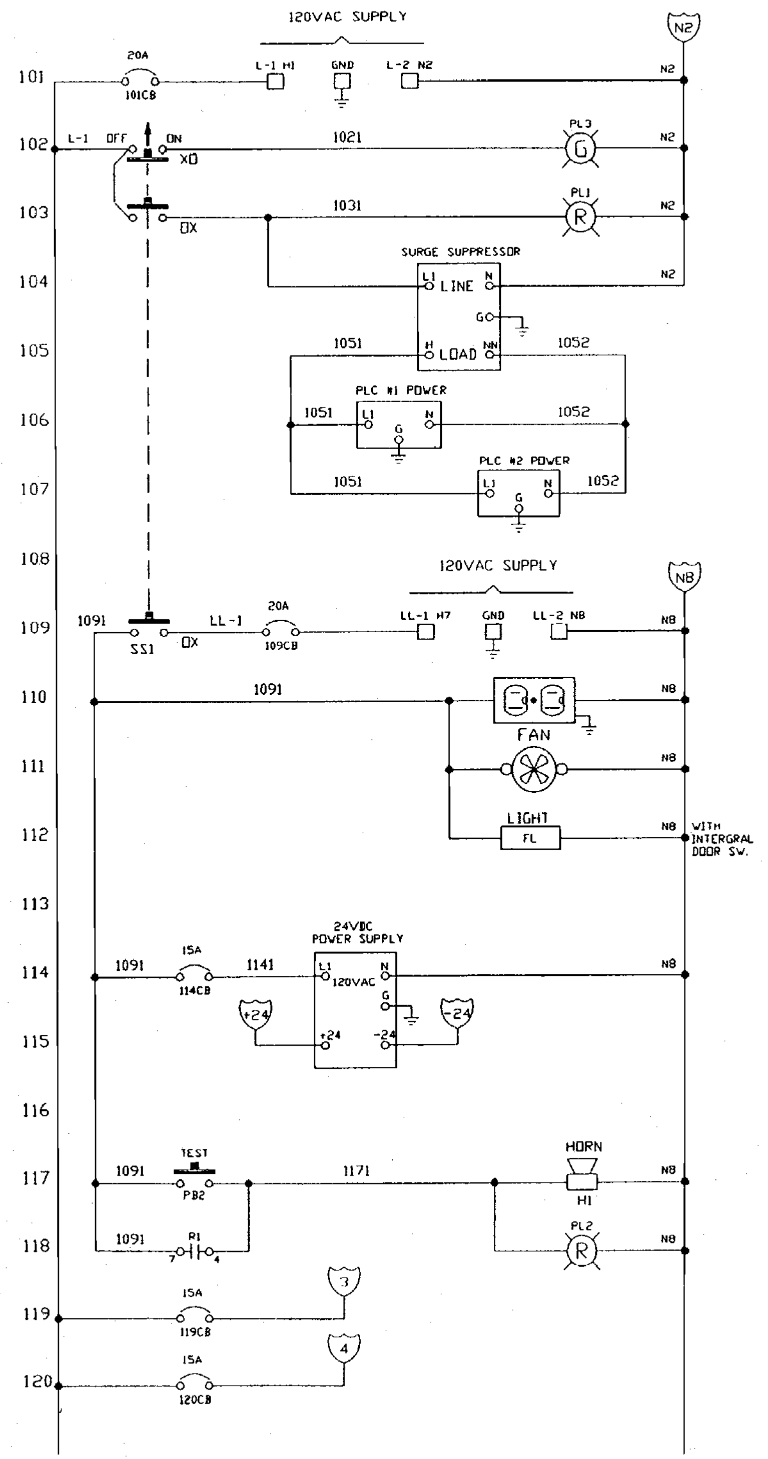
PROCESS CONTROL SERVICES, LTD
P.O. BOX 98 SEAFORD, VIRGINIA

REV NO	DR.	CHK	APP	APP	DATE

DR. RD 05/95
CHK
APP.
APP.
CADD NO.
ASSEMBLY
REFERENCE

ATLANTIC DIVISION
SOIL AND GROUNDWATER REMEDIATION
RECOVERY WELL SYSTEM
PLC CONTROL PANEL
PANEL LAYOUT

SCALE
MICRO
SIZE D
DWG NO. PC9506 1/7



PLC-1

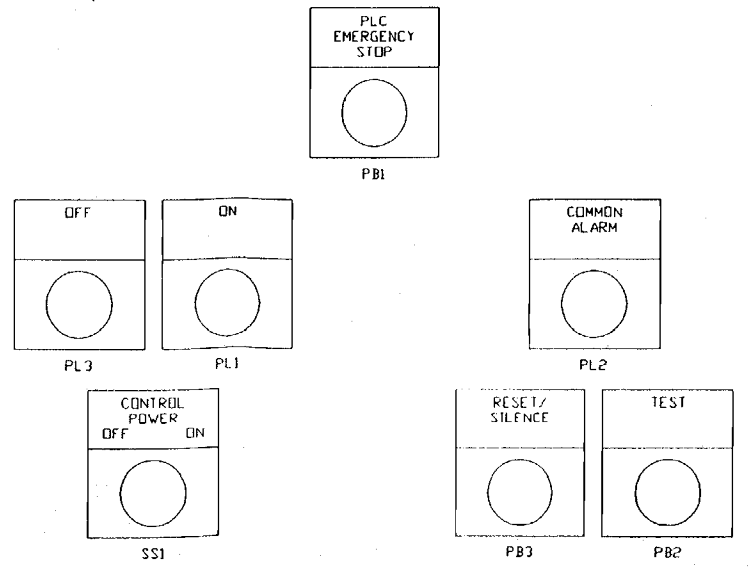
SLOT	
	POWER SUPPLY IC697PWR710
1	CPU IC697CPU772
2	TRANDMITTER IC697BEM713
3	DI IC697MDL250
4	DI IC697MDL250
5	DO IC697MDL341
6	DO IC697MDL341
7	DO IC697MDL341
8	SPACE
9	SPACE

RACK IC697CHS790

PLC-2

SLOT	
	POWER SUPPLY IC697PWR710
1	RECEIVER IC697BEM711
2	AI IC697ALG230
3	AI IC697ALG230
4	AI IC697ALG230
5	AI IC697ALG230
6	AI IC697ALG230
7	AI IC697ALG230
8	AO IC697ALG320
9	AO IC697ALG320

RACK IC697CHS790

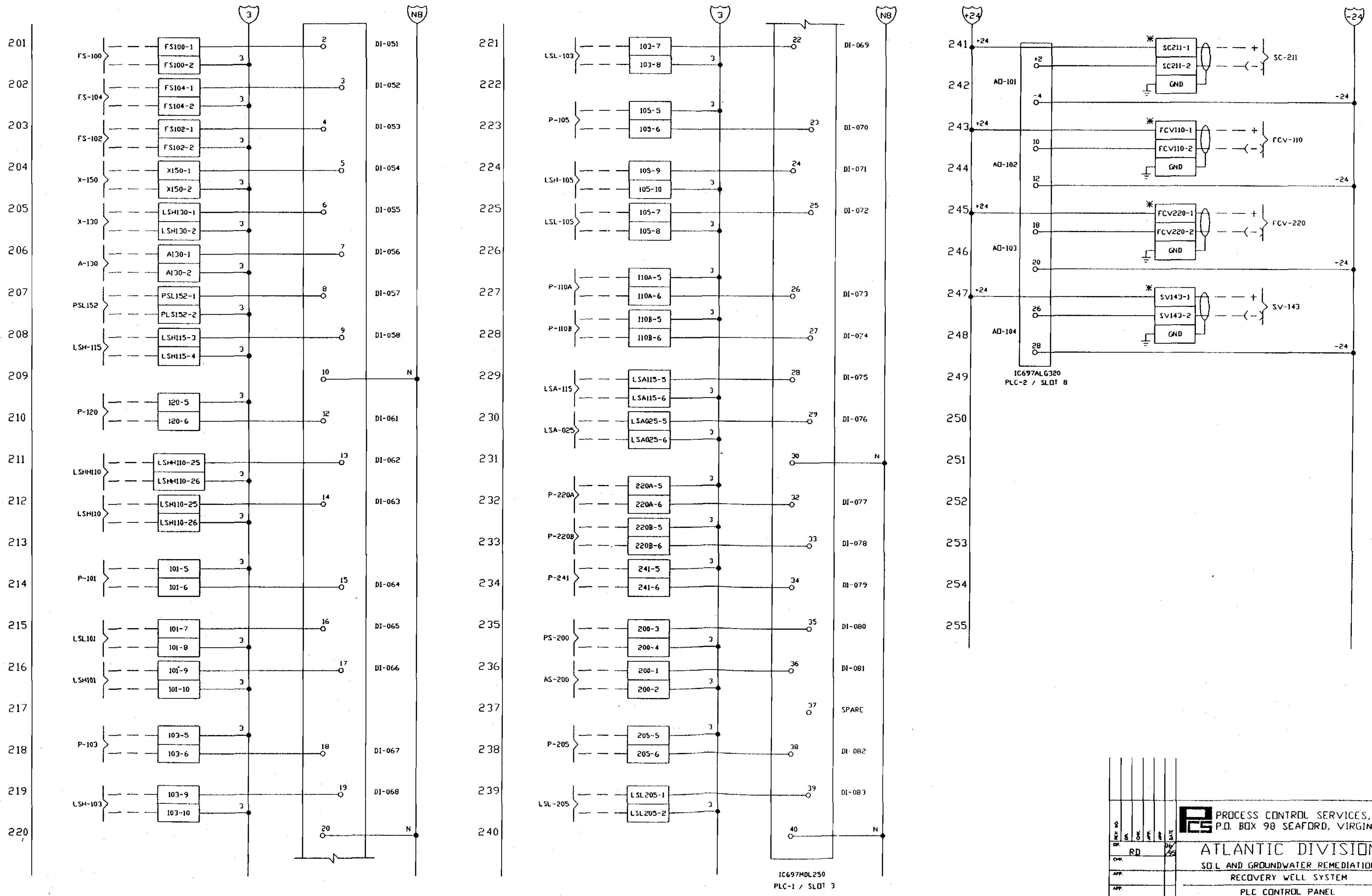


LEGEND PLATES

PROCESS CONTROL SERVICES, LTD
P.O. BOX 98 SEAFORD, VIRGINIA

ATLANTIC DIVISION
SOIL AND GROUNDWATER REMEDIATION
RECOVERY WELL SYSTEM
PLC CONTROL PANEL
POWER DIAGRAM & PLC LAYOUT

DR. RD

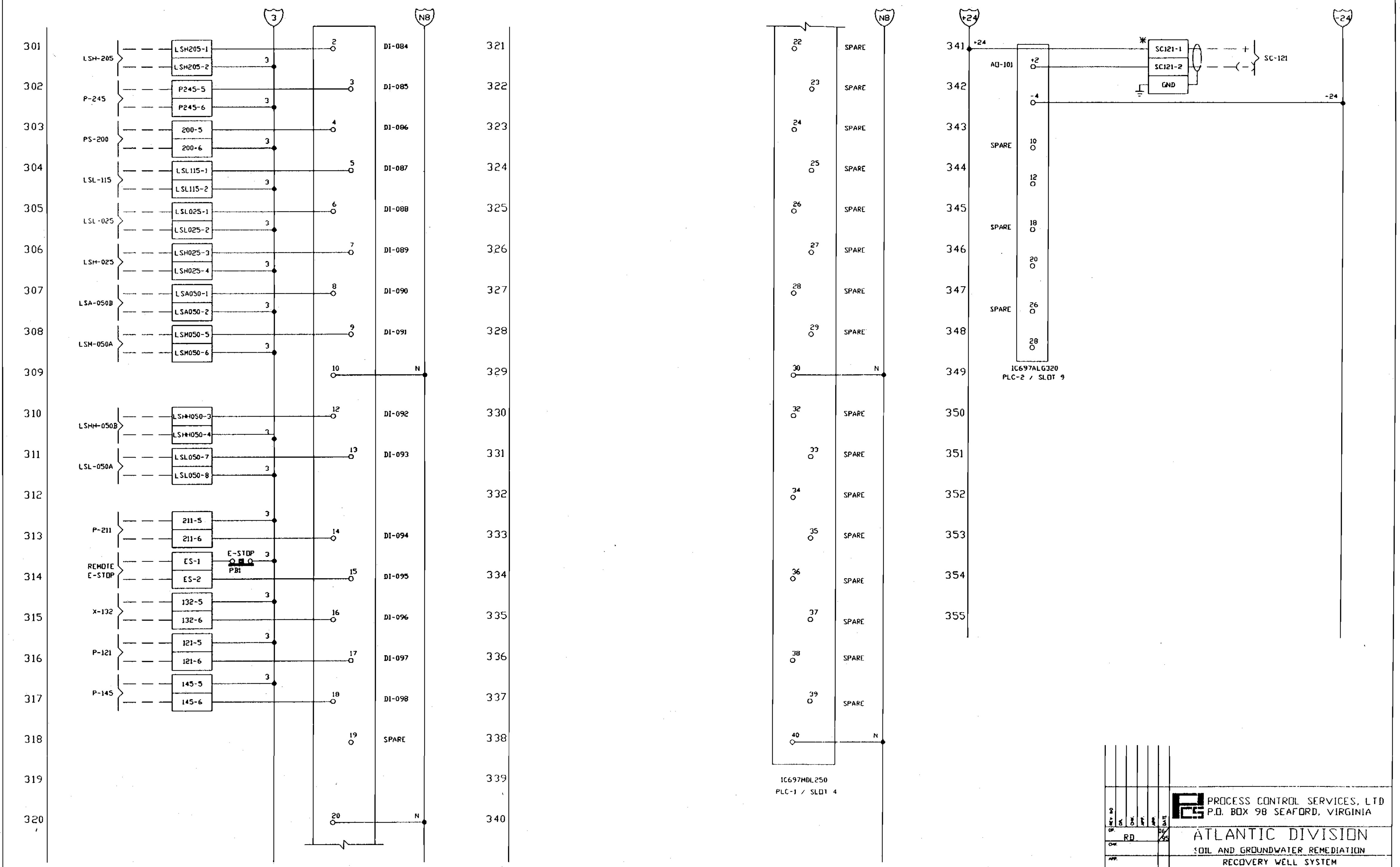


* = FUSED TERM. BLOCK

PROCESS CONTROL SERVICES, LTD
P.O. BOX 98 SEAFORD, VIRGINIA

ATLANTIC DIVISION
SOIL AND GROUNDWATER REMEDIATION
RECOVERY WELL SYSTEM
PLC CONTROL PANEL
WIRING DIAGRAM / D. INPUT & A. OUTPUTS

DR. RD



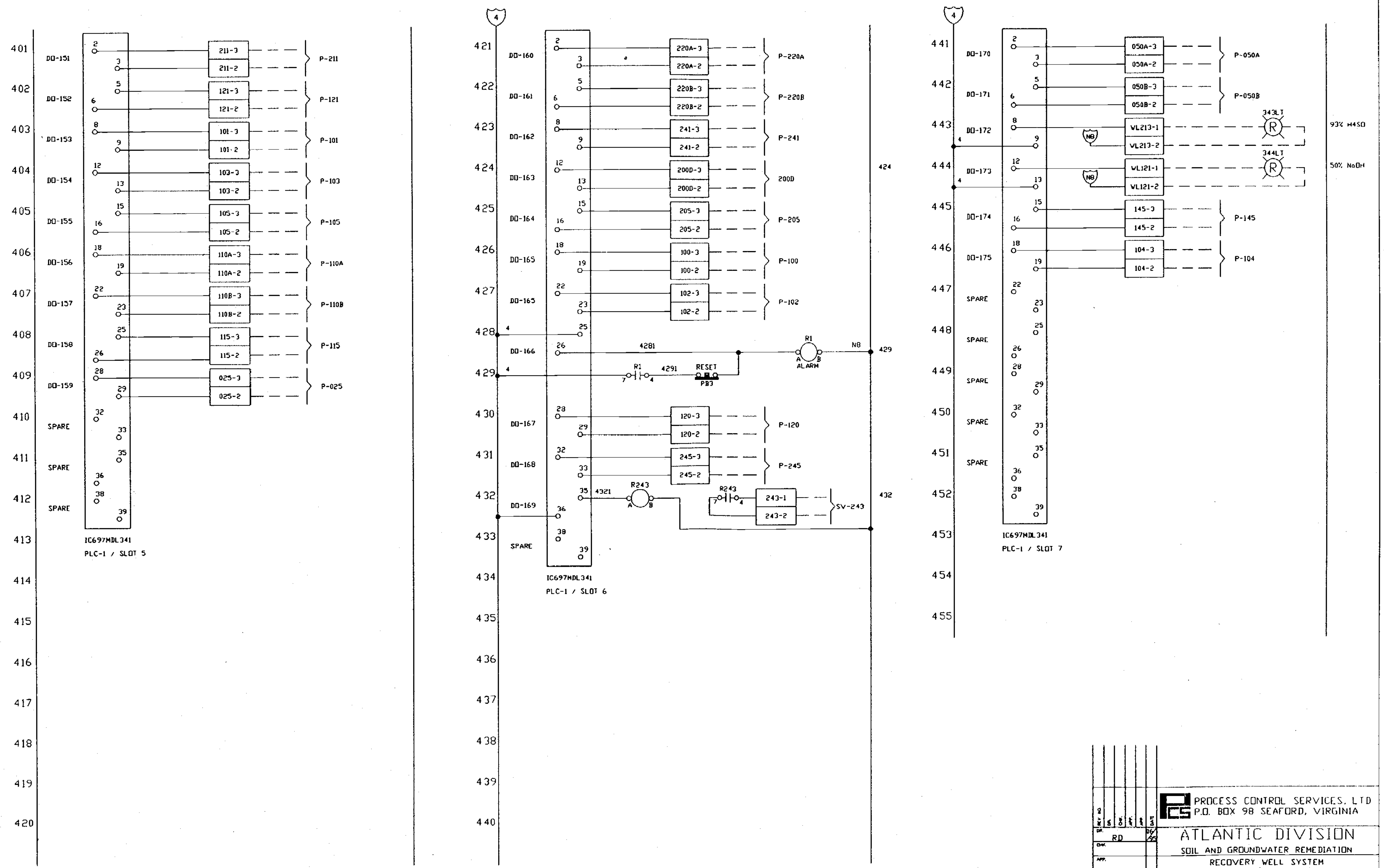
301 LSH-205 LSH205-1 LSH205-2 DI-084
302 P-245 P245-5 P245-6 DI-085
303 PS-200 200-5 200-6 DI-086
304 LSL-115 LSL115-1 LSL115-2 DI-087
305 LSL-025 LSL025-1 LSL025-2 DI-088
306 LSH-025 LSH025-3 LSH025-4 DI-089
307 LSA-050B LSA050-1 LSA050-2 DI-090
308 LSH-050A LSH050-5 LSH050-6 DI-091
309 N
310 LSHH-050B LSHH050-3 LSHH050-4 DI-092
311 LSL-050A LSL050-7 LSL050-8 DI-093
312
313 P-211 211-5 211-6 DI-094
314 REMOTE E-STOP ES-1 ES-2 E-STOP PB1 DI-095
315 X-132 132-5 132-6 DI-096
316 P-121 121-5 121-6 DI-097
317 P-145 145-5 145-6 DI-098
318 SPARE
319
320 N
321 – 340
SPARE
IC697MDL250
PLC-1 / SLOT 4
341 +24
342 AD-101
SPARE
SPARE
SPARE
IC697ALG320
PLC-2 / SLOT 9
SC121-1
SC121-2
GND
SC-121
-24

* = FUSED TERM . BLOCK

PROCESS CONTROL SERVICES, LTD
P.O. BOX 98 SEAFORD, VIRGINIA

ATLANTIC DIVISION
SOIL AND GROUNDWATER REMEDIATION
RECOVERY WELL SYSTEM
PLC CONTROL PANEL
WIRING DIAGRAM / D. INPUT & A. OUTF

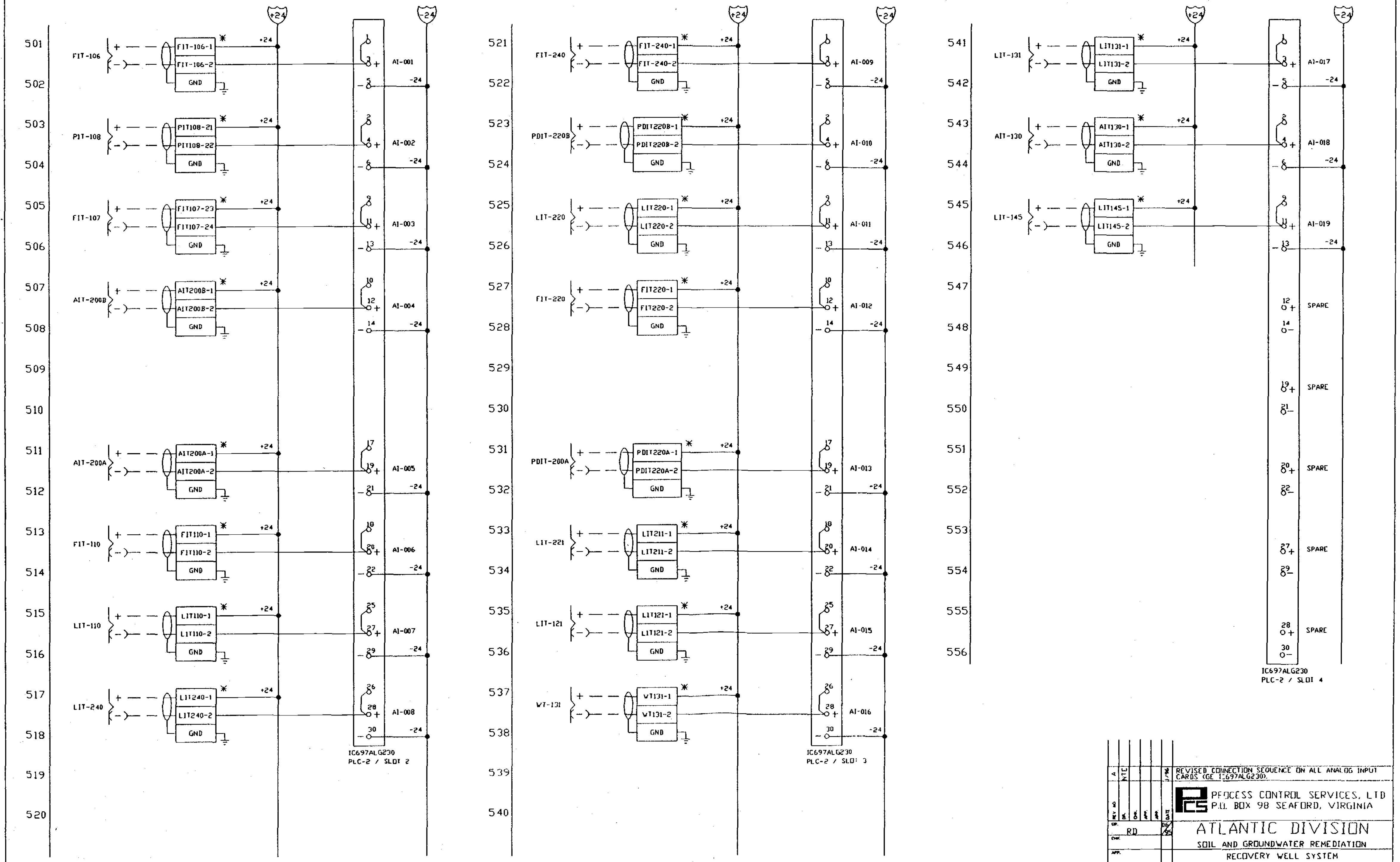
RD


401 DO-151 211-3 211-2 P-211
402 DO-152 121-3 121-2 P-121
403 DO-153 101-3 101-2 P-101
404 DO-154 103-3 103-2 P-103
405 DO-155 105-3 105-2 P-105
406 DO-156 110A-3 110A-2 P-110A
407 DO-157 110B-3 110B-2 P-110B
408 DO-158 115-3 115-2 P-115
409 DO-159 025-3 025-2 P-025
410 SPARE
411 SPARE
412 SPARE
413
414
415
416
417
418
419
420
IC697MDL341
PLC-1 / SLOT 5
421 DO-160 220A-3 220A-2 P-220A
422 DO-161 220B-3 220B-2 P-220B
423 DO-162 241-3 241-2 P-241
424 DO-163 200D-3 200D-2 200D
425 DO-164 205-3 205-2 P-205
426 DO-165 100-3 100-2 P-100
427 DO-165 102-3 102-2 P-102
428
429 DO-166 4281 R1 ALARM N8 4291 RESET PB3
430 DO-167 120-3 120-2 P-120
431 DO-168 245-3 245-2 P-245
432 DO-169 4321 R243 243-1 243-2 SV-243
433 SPARE
434
435
436
437
438
439
440
IC697MDL341
PLC-1 / SLOT 6
441 DO-170 050A-3 050A-2 P-050A
442 DO-171 050B-3 050B-2 P-050B
443 DO-172 VL213-1 VL213-2 343LT 93% H4SO
444 DO-173 VL121-1 VL121-2 344LT 50% NaOH
445 DO-174 145-3 145-2 P-145
446 DO-175 104-3 104-2 P-104
447 SPARE
448 SPARE
449 SPARE
450 SPARE
451 SPARE
452
453
454
455
IC697MDL341
PLC-1 / SLOT 7

PROCESS CONTROL SERVICES, LTD
P.O. BOX 98 SEAFORD, VIRGINIA

DR. RD

ATLANTIC DIVISION
SOIL AND GROUNDWATER REMEDIATION
RECOVERY WELL SYSTEM
PLC CONTROL PANEL
WIRING DIAGRAM / DISCRETE OUTPU



✱ = FUSED TERMINAL ƆCK

A	NTE				11/96	REVISED CONNECTION SEQUENCE ON ALL ANALOG INPUT CARDS (GE IC697ALG230).
REV NO	DR.	CHK.	APP.	APP.	DATE	PROCESS CONTROL SERVICES, LTD P.O. BOX 98 SEAFORD, VIRGINIA

DR. RD 06/95

CHK.

APP.

APP.

CADD NO.

ASSEMBLY

REFERENCE

ATLANTIC DIVISION

SOIL AND GROUNDWATER REMEDIATION

RECOVERY WELL SYSTEM

PLC CONTROL PANEL

WIRING DIAGRAM / ANALOG INPUTS

SCALE

MICRO

SIZE D DWG. NO. PC9506 6/7 REV. A

Wiring and Control Diagrams

Maintenance and Repair Procedures

Fuse Chart

Analog Loop	500 mA	GMA-500mA
5 VDC Power Supply	½ Amp	KTK- ½
24 VDC 1.2 Amp Power Supply	½ Amp	KTK- ½
24 VDC 4.8 Amp Power Supply	2 Amps	KTK- 2

Spare Parts and Supply List

Replacement equipment for components in the PLC system are available either directly from the manufacturer, their sales representative, or an integration/systems design company. Below is a list of components contained in the PLC system. This list also shows sources of supply for these components. If expert help is needed, this system was manufacturer by:

Process Control Services, Ltd.
P.O. Box 98
Seaford, VA 23696
(757) 898-4332
FAX (757) 898-8625

PLC Back plane	IC697CHS790	<u>Manufacturer</u>
PLC CPU	IC697CPU772	GE Fanuc
PLC Power Supply	IC697PWR711	P.O. Box 8106
PLC Memory Module	IC697MEM713	Charlottesville, VA 22906
PLC Ethernet Module	IC697CMM741	(800) 828-5747
PLC Bus Transmitter	IC697BEM713	
PLC Bus Receiver	IC697BEM711	<u>Supply</u>
PLC Digital Input Mod.	IC697MDL250	Contact manufacturer.
PLC Digital Output Mod.	IC697MDL341	
PLC Relay Output Mod.	IC697MDL940	
PLC Analog Input Mod.	IC697ALG230	
PLC Analog Output Mod.	IC697ALG320	
Lithium Battery	IC697ACC702	

Telemetry Output Card	T2021F02	<u>Manufacturer</u>
Telemetry Input Card	T2021E01	D Squared Services
Line Logic Low Card	A1036C01	4362 Rosebrier Ave.
Telemetry Master	T2033A01	White Bear Lake, MN 55127
Telemetry Motherboard	T2034B01	(612) 653-6242
		<u>Supply</u>
		Process Control Services, Ltd.

Bridging Amplifier Telemetry Remote Term.	BA200 RT100	<u>Manufacturer</u> Crosstar 1780 Venus Ave. N. Arden Hills, MN 55112 (612) 636-8827 <u>Supply</u> Process Control Services, Ltd.
5 VDC Power Supply 24 VDC Power Supply 24 VDC Power Supply	IHA5-1.2/OVP IHD24-4.8 IHB24-1.2	<u>Manufacturer</u> International Power 360 Bernoulli Circle Oxnard, CA 93030 (800) 845-5386 <u>Supply</u> Contact Manufacturer.
Power Supply Assembly	OLD-512AA	<u>Manufacturer</u> TRC Electronics 135 Pasadena Ave. Lodi, NJ 07644 (201) 779-8282 <u>Supply</u> Contact Manufacturer.
Replacement Bulbs Replacement PLC Key	BA9S615 077C3095	<u>Manufacturer</u> GE Electrical Distribution and Control 4601 Park Rd. Suite 400 Charlotte, NC 28209 (800) 431-7867 <u>Supply</u> Contact Manufacturer.

Warranty Information

***** L O G I C T A B L E O F C O N T E N T S *****

```
|[  START OF LD  PROGRAM PC9506    ]        (*                                          *)
|
|      VARIABLE DECLARATIONS       ]
```

V A R I A B L E   D E C L A R A T I O N   T A B L E

REFERENCE	NICKNAME	REFERENCE DESCRIPTION
%I00081	P100S	SRW1 RUN STATUS
%I00082	P102S	SRW2 RUN STATUS
%I00083	P104S	SRW3 RUN STATUS
%I00084	LSL220	STRIP TANK LO
%I00085	LSH220	STRIP TANK HI
%I00086	A130S	FLOCC MIXER RUN STATUS
%I00087	PSL152	AIR SYSTEM LOW PRESS
%I00088	LSH115	CONTAIN AREA SUMP HIGH
%I00089	P120S	JET MIX PUMP RUN STATUS
%I00090	LSHH110	GROUND WATER TANK HI-HI
%I00091	LSH110	GROUND WATER TANK HI
%I00092	P101S	DRW1 RUN STATUS
%I00093	LSL101	DRW1 LLCO
%I00094	LSL110	GROUND WATER TANK LO
%I00095	P103S	DRW2 RUN STATUS
%I00097	LSL103	DRW2 LLCO
%I00098	P105S	DRW3 RUN STATUS
%I00100	LSL105	DRW3 LLCO
%I00101	P110AS	STRIP PUMP RUN STATUS
%I00102	P110BS	STRIP PUMP RUN STATUS
%I00103	LSA115	CONTAIN AREA SUMP ALARM
%I00104	LSA025	BUILD DRAIN SUMP ALARM
%I00105	P220AS	GAC FEED PUMP STATUS
%I00106	P220BS	GAC FEED PUMP STATUS
%I00107	P241S	BACK WASH PUMP STATUS
%I00108	PSH200	COLUMN FAN DISCH LOW
%I00109	K200S	COLUMN FAN RUN STATUS
%I00111	P205S	SPENT BACK WASH PUMP
%I00112	LSL205	BACK WASH HOLDING TANK
%I00113	LSH205	BACK WASH HOLDING TANK
%I00114	P245S	REUSE WATER PUMP STATUS
%I00115	PSL200	COLUMN FAN DISCH HIGH
%I00116	LSL115	CONTAIN AREA SUMP LOW
%I00117	LSL025	BUILD DRAIN SUMP LOW
%I00118	LSH025	BUILD DRAIN SUMP HIGH
%I00124	ESTOP	REMOTE EMERG STOP
%I00127	P145S	TRANSFR PUMP RUN STATUS
%I00128	LSL100	SRW1 LLCO
%I00129	LSL300	SRW4 LLCO
%I00130	LSL102	SRW2 LLCO
%I00131	LSL302	SRW5 LLCO
%I00132	LSL104	SRW3 LLCO
%I00133	LSL304	SRW6 LLCO
%I00134	P300S	SRW4 RUN STATUS
%I00135	P302S	SRW5 RUN STATUS
%I00136	P304S	SRW6 RUN STATUS

%I00137	P301S	DMW1 RUN STATUS
%I00138	LSL301	DMW1 LLCO
%Q00001	P211C	ACID PUMP CALL
%Q00002	P121C	CAUSTIC PUMP CALL
%Q00003	P101C	DRW-1 PUMP CALL
%Q00004	P103C	DRW-2 PUMP CALL
%Q00005	P105C	DRW-3 PUMP CALL
%Q00006	P110AC	STRIP FEED PUMP CALL
%Q00007	P110BC	STRIP FEED PUMP CALL
%Q00008	P115C	CONTAIN SUMP PUMP CALL
%Q00009	P025C	BUILD DRAINS PUMP CALL
%Q00010	P300C	SRW4 PUMP CALL
%Q00011	P302C	SRW5 PUMP CALL
%Q00012	P304C	SRW6 PUMP CALL
%Q00017	P220AC	GAC FEED PUMP CALL
%Q00018	P220BC	GAC FEED PUMP CALL
%Q00019	P241C	BACK WASH PUMP CALL
%Q00020	K200C	COLUMN FAN CALL
%Q00021	P205C	SPENT WASH PUMP CALL
%Q00022	P100C	SRW-1 PUMP CALL
%Q00023	P102C	SRW-2 PUMP CALL
%Q00024	ALARM	COMMON ALARM
%Q00025	P120C	JET MIX PUMP CALL
%Q00026	P245C	REUSE WATER PUMP CALL
%Q00027	X132C	POLYMER SYSTEM
%Q00028	P212C	SPARE ACID PUMP
%Q00033	P050AC	WW PUMP A CALL
%Q00034	P050BC	WW PUMP B CALL
%Q00035	Y343LT	ACID TANK FULL
%Q00036	Y344LT	CAUSTIC TANK FULL
%Q00037	P145C	TRANSFR PUMP CALL
%Q00038	P104C	SRW-3 PUMP CALL
%Q00039	P301C	DMW1 PUMP CALL
%M00001	ALLSTOP	EMERGENCY STOP HOLD BIT
%M00002	RESET	EMERGENCY STOP HOLD RESET
%M00027	LCM145	P-145 ON CONTROL
%M00028	FAL106	SRW LO FLOW
%M00029	FAL107	DRW LO FLOW
%M00030	LAH130	
%M00031	LAL205	BK WASH HOLDING TK LOW LEVEL
%M00032	LAH205	BK WASH HOLDING TK HIGH LEVEL
%M00033	PAL150	AIR COMPR LOW PRESS
%M00034	LAL240	TREATED EFFL TK LOW LEVEL
%M00035	LALL240	TREATED EFFL TK LO-LO LEVEL
%M00036	PDH220A	CART FILTER HI D/P
%M00037	PDH220B	GAC FILTER HI D/P
%M00038	FAH220	FILTER FEED HI FLOW
%M00039	FAL220	FILTER FEED LOW FLOW
%M00040	PAH200	STRIP BLOWER HI PRESS
%M00041	PAL200	STRIP BLOWER LO PRESS
%M00042	LAH025	BLDG DR HI LEVEL
%M00043	LAL100	SRW1 LO LEVEL
%M00045	LAL101	DRW1 LO LEVEL
%M00047	LAL102	SRW2 LO LEVEL

%M00049	LAL103	DRW2 LO LEVEL
%M00051	LAL104	SRW3 LO LEVEL
%M00053	LAL105	DRW3 LO LEVEL
%M00055	LAL300	SRW4 LLCO
%M00056	LAH106	NEW PUMP
%M00057	LAL302	SRW5 LLCO
%M00058	LAH107	NEW PUMP
%M00059	LAL304	SRW6 LLCO
%M00060	LAH108	NEW PUMP
%M00061	LAL301	DMW1 LLCO
%M00062	LAH109	NEW PUMP
%M00072	LAHH110	GROUND WATER TANK HI-HI
%M00073	LAH110	GROUND WATER TANK HI
%M00074	LAL110	GROUND WATER TANK LO
%M00076	WIA131	POLYMER TANK LOW ALARM
%M00077	LAH115	CONTAIN SUMP HI LEVEL
%M00079	PAL108	DRW LOW PRESS
%M00080	AAH130	MIX TK HIGH PH
%M00081	AAL130	MIX TK LOW PH
%M00082	LAH121	CAUSTIC TANK HIGH LEVEL
%M00083	LAL121	CAUSTIC TANK LOW LEVEL
%M00084	LAH131	CLARIF HIGH LEVEL
%M00085	LAH145	HEAD TANK HIGH LEVEL
%M00086	LAL145	HEAD TANK LOW LEVEL
%M00089	FAL110	STRIP FEED LO FLOW
%M00090	FAH110	STRIP FEED HIGH FLOW
%M00091	LAH211	ACID TANK HIGH LEVEL
%M00092	LAL211	ACID TANK LOW LEVEL
%M00093	LAHH220	STRIP TANK HI-HI LEVEL
%M00094	LAH220	STRIP TANK HI LEVEL
%M00095	LAL220	STRIP TANK LO LEVEL
%M00096	AAH200	T-110 HIGH PH
%M00097	AAL200	T-110 LOW PH
%M00098	LAHH240	TREATED EFFL TK HI-HI LEVEL
%M00099	LAH240	TREATED EFFL TK HI LEVEL
%M00100	I100	
%M00101	I101	
%M00102	I102	
%M00103	I103	
%M00104	I104	
%M00105	I105	
%M00106	I300	NEW PUMP
%M00107	I301	NEW PUMP
%M00108	I302	NEW PUMP
%M00109	I304	NEW PUMP
%M00110	I110	
%M00111	I111	
%M00121	I121	
%M00122	I122	
%M00123	I123	
%M00130	I130	
%M00132	I132	
%M00145	I145	
%M00146	I146	

```
%M00150     I150
%M00151     I151
%M00152     I152
%M00200     I200
%M00201     I201
%M00202     I202
%M00205     I205
%M00206     I206
%M00211     I211
%M00212     I212
%M00213     I213
%M00220     I220
%M00221     I221
%M00222     I222
%M00223     I223
%M00240     I240
%M00242     I242
%M00301     P025X       FORCE ON
%M00306     P025I       START INTLK
%M00307     P025D       STOP INTLK
%M00308     P100X       FORCE ON
%M00313     P100I       START INTLK
%M00314     P100D       STOP INTLK
%M00315     P101X       FORCE ON
%M00320     P101I       START INTLK
%M00321     P101D       STOP INTLK
%M00322     P102X       FORCE ON
%M00327     P102I       START INTLK
%M00328     P102D       STOP INTLK
%M00329     P103X       FORCE ON
%M00334     P103I
%M00335     P103D
%M00336     P104X
%M00341     P104I
%M00342     P104D
%M00343     P105X
%M00348     P105I
%M00349     P105D
%M00350     P300X       NEW PUMP
%M00355     P300I       "
%M00356     P300D       "
%M00357     P302X       NEW PUMP
%M00362     P302I       "
%M00363     P302D       "
%M00364     P304X       NEW PUMP
%M00369     P304I       "
%M00370     P304D       "
%M00371     P301X       NEW PUMP
%M00376     P301I       "
%M00377     P301D       "
%M00378     P110AX
%M00383     P110AI
%M00384     P110AD
%M00385     P110BX
```

```
%M00390     P110BI
%M00391     P110BD
%M00392     P115X
%M00397     P115I
%M00398     P115D
%M00399     P120X
%M00404     P120I
%M00405     P120D
%M00406     P121X
%M00411     P121I
%M00412     P121D
%M00413     A130X
%M00418     A130I
%M00419     A130D
%M00420     X132X
%M00425     X132I
%M00426     X132D
%M00427     P145X
%M00432     P145I
%M00433     P145D
%M00434     K200X
%M00439     K200I
%M00440     K200D
%M00441     P205X
%M00446     P205I
%M00447     P205D
%M00448     P211X
%M00453     P211I
%M00454     P211D
%M00455     P220AX
%M00460     P220AI
%M00461     P220AD
%M00462     P220BX
%M00467     P220BI
%M00468     P220BD
%M00469     P241X
%M00474     P241I
%M00475     P241D
%M00476     P245X
%M00481     P245I
%M00482     P245D
%M00483     X150X
%M00488     X150I
%M00489     X150D
%M00491     SV243X
%M00496     SV243I
%M00497     SV243D
%M00512     AY130A
%M00513     AY130B
%M00514     AY130C
%M00515     AX130A
%M00516     AX130B
%M00517     AX130C
%M00518     AY200A
```

```
%M00519      AY200B
%M00520      AY200C
%M00521      AX200A
%M00522      AX200B
%M00523      AX200C
%R00001      FI106        SHALLOW WELLS COMB FLOW RATE
%R00002      FQ106        SHALLOW WELLS ACCUM FLOW
%R00003      FI107        DEEP WELLS COMB FLOW RATE
%R00004      FQ107        DEEP WELLS ACCUM FLOW
%R00005      PI108
%R00008      LI131
%R00009      WI132
%R00010      LI145
%R00011      AI130A
%R00012      AI130B
%R00013      AI130C
%R00014      LI121
%R00015      AI200A
%R00016      AI200B
%R00017      AI200C
%R00018      LI211
%R00019      PD220A
%R00020      PD220B
%R00021      LI240
%R00022      FI240        EFFLUENT FLOW RATE
%R00023      FQ240        EFFLUENT ACCUM FLOW
%R00025      FQ106_D      FQ106
%R00027      FQ107_D      FQ107
%R00029      FQ240_D      FQ240
%R00031      FI106_D      FI106 AS DINT
%R00033      FI107_D      FI107 AS DINT
%R00035      FI240_D      FI240 AS DINT
%R00100      L240_LL
%R00101      P108_L
%R00102      L110_L
%R00103      F110_L
%R00104      W131_L
%R00105      L145_L
%R00106      A130_L
%R00107      L121_L
%R00108      A200_L
%R00109      L211_L
%R00110      F220_L
%R00111      L220_L
%R00112      L240_L
%R00113      L110_H
%R00114      F110_H
%R00115      L131_H
%R00116      L145_H
%R00117      A130_H
%R00118      L121_H
%R00119      A200_H
%R00120      L211_H
%R00121      P220A_H
```

%R00122	P220B_H	
%R00123	F220_H	
%R00124	L220_H	
%R00125	L240_H	
%R00126	L220_HH	
%R00127	L240_HH	
%R00128	L110_HH	
%R00129	F106_L	
%R00130	F107_L	
%R00131	L145_M	
%R00200	L110_S	SETPOINT
%R00201	LI110	PROCESS VARIABLE
%R00202	F110_S	CV FROM LC110
%R00203	FI110	PROCESS VARIABLE
%R00204	F110_C	CV FROM FC110 TO FCV110
%R00205	L220_S	SETPOINT
%R00206	LI220	PROCESS VARIABLE
%R00207	F220_S	CV FROM LC220
%R00208	FI220	PROCESS VARIABLE
%R00209	F220_C	CV FROM FC220 TO FCV220
%R00210	A130_S	SETPOINT
%R00211	AI130	PROCESS VARIABLE
%R00212	A130_C	CV TO SC121
%R00213	A200_S	SETPOINT
%R00214	AI200	PROCESS VARIABLE
%R00215	A200_C	CV TO SC211
%R00300	A130SEL	A130 SELECT
%R00301	A200SEL	A200 SELECT
%R00302	TMR_240	LOW T240 LEVEL
%R00305	TMR_220	K200 SHUTDOWN
%R00308	TMR_221	K200 SHUTDOWN
%R00400	LC110	PID 1 LEVEL CONTROL
%R00440	FC110	PID 2 FLOW CONTROL
%R00480	LC220	PID 3 LEVEL CONTROL
%R00520	FC220	PID 4 FLOW CONTROL
%R00560	AC130	PID 5 PH CONTROL
%R00600	AC200	PID 6 PH CONTROL
%AI0001	FIT106	SHALLOW WELLS COMB FLOW
%AI0002	PIT108	DEEP WELLS COMB PRESS
%AI0003	FIT107	DEEP WELLS COMB FLOW
%AI0004	AIT200B	T-110 RETURN PH
%AI0005	AIT200A	T-110 RETURN PH
%AI0006	FIT110	WATER TO AIR STRIP FLOW
%AI0007	LIT110	GROUND WATER TANK LEVEL
%AI0008	LIT240	TREATED EFFLU TANK LEVEL
%AI0009	FIT240	TREATED EFFL TO WALLACE CREEK
%AI0010	PDT220B	GAC ABSORB D/P
%AI0011	LIT220	AIR STRIP WATER LEVEL
%AI0012	FIT220	CRTDG FILTER FLOW
%AI0013	PDT220A	CRTDG FILTER D/P
%AI0014	LIT211	ACID STORAGE TANK LEVEL
%AI0015	LIT121	CAUSTIC STORAGE TANK LEVEL
%AI0016	WT131	POLYMER DRUM WEIGHT
%AI0017	LIT131	CLARIF BUBBLER LEVEL

%AI0018	AIT130A	MIX TNK CHAMB 1 PH
%AI0019	LIT145	HEAD TANK LEVEL
%AI0020	AIT130B	MIX TNK CHAMB 2 PH
%AI0021	AIT130C	MIX TNK CHAMB 3 PH
%AI0022	AIT200C	STRIP PUMP SUCTION PH
%AQ0001	SC211	ACID PUMP STROKE CONTROL
%AQ0002	FCV110	PROCESS WATER TO AIR STRIP
%AQ0003	FCV220	GAC ABSORB FEED FLOW
%AQ0005	SC121	CAUSTIC PUMP STROKE CONTROL

I D E N T I F I E R   T A B L E

IDENTIFIER	IDENTIFIER TYPE	IDENTIFIER DESCRIPTION
INTLKS	PROGRAM BLOCK	DEFINED INTERLOCKS
E_STOP	PROGRAM BLOCK	EMERGENCY STOP ROUTINE
PH	PROGRAM BLOCK	
POLYMER	PROGRAM BLOCK	POLYMER FEED SYSTEM
FLOWS	PROGRAM BLOCK	
PRESS	PROGRAM BLOCK	PRESSURE
AIR	PROGRAM BLOCK	INSTREUMENT AIR COMPRESSOR
LEVEL	PROGRAM BLOCK	
PUMPS	PROGRAM BLOCK	
MISC	PROGRAM BLOCK	
P_100	PROGRAM BLOCK	SHALLOW WELL 1
P_101	PROGRAM BLOCK	DEEP WELL 1
P_102	PROGRAM BLOCK	SHALLOW WELL 2
P_103	PROGRAM BLOCK	DEEP WELL 2
P_104	PROGRAM BLOCK	SHALLOW WELL 3
P_105	PROGRAM BLOCK	DEEP WELL 3
P_025	PROGRAM BLOCK	BUILDING DRAIN SUMP
P_115	PROGRAM BLOCK	TANKER OVERFLOW SUMP
P_121	PROGRAM BLOCK	CAUSTIC PUMP
P_145	PROGRAM BLOCK	PARTIALLY TREATED WATER
P_211	PROGRAM BLOCK	ACID PUMP
P_212	PROGRAM BLOCK	SPARE ACID PUMP
P_120	PROGRAM BLOCK	JET MIXING PUMP
P_110A	PROGRAM BLOCK	AIR STRIPPER FEED
P_110B	PROGRAM BLOCK	AIR STRIPPER FEED
P_205	PROGRAM BLOCK	SPENT BACKWASH WATER
P_220A	PROGRAM BLOCK	GAC ADSORBER FEED
P_220B	PROGRAM BLOCK	GAC ADSORBER FEED
P_245	PROGRAM BLOCK	REUSE WATER
P_241	PROGRAM BLOCK	BACKWASH WATER
P_300	PROGRAM BLOCK	SRW 4
P_301	PROGRAM BLOCK	DMW 1
P_302	PROGRAM BLOCK	SRW 5
P_304	PROGRAM BLOCK	SRW 6
K_200	PROGRAM BLOCK	STRIPPER TOWER BLOWER
FI_106	PROGRAM BLOCK	SHALLOW WELL COMB FLOW
FQ_106	PROGRAM BLOCK	SHALLOW WELL COMB FLOW TOTALIZER
FI_107	PROGRAM BLOCK	DEEP WELL COMB FLOW
FQ_107	PROGRAM BLOCK	DEEP WELL COMB FLOW TOTALIZER

```
          FIC_110          PROGRAM BLOCK       AIR STRIPPER FEED FLOW
          FIC_220          PROGRAM BLOCK       GAC ADSORBER FEED FLOW
          FQ_240           PROGRAM BLOCK       TREATED EFFLUENT TOTALIZER
          LIC_110          PROGRAM BLOCK       GROUND WATER STORAGE TANK
          LIC_121          PROGRAM BLOCK       CAUSTIC STORAGE TANK
          LIC_131          PROGRAM BLOCK       CLARIFIER
          LIC_145          PROGRAM BLOCK       HEAD TANK
          LIC_220          PROGRAM BLOCK       STRIPPER EFFLUENT HOLDING TANK
          LIC_240          PROGRAM BLOCK       TREATED EFFLUENT HOLDING TANK
          LIC_211          PROGRAM BLOCK       ACID STORAGE TANK
          LIC_205          PROGRAM BLOCK       BACKWASH HOLDING TANK
          PC9506           PROGRAM NAME

|[   PROGRAM BLOCK DECLARATIONS    ]

              +--------+
              |INTLKS  |       LANG: LD  (* DEFINED INTERLOCKS                   *)
              +--------+

              +--------+
              |E_STOP  |       LANG: LD  (* EMERGENCY STOP ROUTINE               *)
              +--------+

              +--------+
              |  PH    |       LANG: LD  (*                                      *)
              +--------+

              +--------+
              |POLYMER|        LANG: LD  (* POLYMER FEED SYSTEM                  *)
              +--------+

              +--------+
              | FLOWS  |       LANG: LD  (*                                      *)
              +--------+

              +--------+
              | PRESS  |       LANG: LD  (* PRESSURE                             *)
              +--------+

              +--------+
              |  AIR   |       LANG: LD  (* INSTREUMENT AIR COMPRESSOR           *)
              +--------+

              +--------+
              | LEVEL  |       LANG: LD  (*                                      *)
              +--------+

              +--------+
              | PUMPS  |       LANG: LD  (*                                      *)
              +--------+

              +--------+
              | MISC   |       LANG: LD  (*                                      *)
              +--------+
+--------+
| P_100 |        LANG: LD   (* SHALLOW WELL 1                        *)
+--------+

+--------+
| P_101 |        LANG: LD   (* DEEP WELL 1                           *)
+--------+

+--------+
| P_102 |        LANG: LD   (* SHALLOW WELL 2                        *)
+--------+

+--------+
| P_103 |        LANG: LD   (* DEEP WELL 2                           *)
+--------+

+--------+
| P_104 |        LANG: LD   (* SHALLOW WELL 3                        *)
+--------+

+--------+
| P_105 |        LANG: LD   (* DEEP WELL 3                           *)
+--------+

+--------+
| P_025 |        LANG: LD   (* BUILDING DRAIN SUMP                   *)
+--------+

+--------+
| P_115 |        LANG: LD   (* TANKER OVERFLOW SUMP                  *)
+--------+

+--------+
| P_121 |        LANG: LD   (* CAUSTIC PUMP                          *)
+--------+

+--------+
| P_145 |        LANG: LD   (* PARTIALLY TREATED WATER               *)
+--------+

+--------+
| P_211 |        LANG: LD   (* ACID PUMP                             *)
+--------+

+--------+
| P_212 |        LANG: LD   (* SPARE ACID PUMP                       *)
+--------+

+--------+
| P_120 |        LANG: LD   (* JET MIXING PUMP                       *)
+--------+
+--------+
|P_110A |         LANG: LD  (* AIR STRIPPER FEED                     *)
+--------+

+--------+
|P_110B |         LANG: LD  (* AIR STRIPPER FEED                     *)
+--------+

+--------+
| P_205 |         LANG: LD  (* SPENT BACKWASH WATER                  *)
+--------+

+--------+
|P_220A |         LANG: LD  (* GAC ADSORBER FEED                     *)
+--------+

+--------+
|P_220B |         LANG: LD  (* GAC ADSORBER FEED                     *)
+--------+

+--------+
| P_245 |         LANG: LD  (* REUSE WATER                           *)
+--------+

+--------+
| P_241 |         LANG: LD  (* BACKWASH WATER                        *)
+--------+

+--------+
| P_300 |         LANG: LD  (* SRW 4                                 *)
+--------+

+--------+
| P_301 |         LANG: LD  (* DMW 1                                 *)
+--------+

+--------+
| P_302 |         LANG: LD  (* SRW 5                                 *)
+--------+

+--------+
| P_304 |         LANG: LD  (* SRW 6                                 *)
+--------+

+--------+
| K_200 |         LANG: LD  (* STRIPPER TOWER BLOWER                 *)
+--------+

+--------+
|FI_106 |         LANG: LD  (* SHALLOW WELL COMB FLOW                *)
+--------+
+--------+
|FQ_106  |        LANG: LD   (* SHALLOW WELL COMB FLOW TOTALIZER *)
+--------+

+--------+
|FI_107  |        LANG: LD   (* DEEP WELL COMB FLOW               *)
+--------+

+--------+
|FQ_107  |        LANG: LD   (* DEEP WELL COMB FLOW TOTALIZER     *)
+--------+

+--------+
|FIC_110|         LANG: LD   (* AIR STRIPPER FEED FLOW            *)
+--------+

+--------+
|FIC_220|         LANG: LD   (* GAC ADSORBER FEED FLOW            *)
+--------+

+--------+
|FQ_240  |        LANG: LD   (* TREATED EFFLUENT TOTALIZER        *)
+--------+

+--------+
|LIC_110|         LANG: LD   (* GROUND WATER STORAGE TANK         *)
+--------+

+--------+
|LIC_121|         LANG: LD   (* CAUSTIC STORAGE TANK              *)
+--------+

+--------+
|LIC_131|         LANG: LD   (* CLARIFIER                         *)
+--------+

+--------+
|LIC_145|         LANG: LD   (* HEAD TANK                         *)
+--------+

+--------+
|LIC_220|         LANG: LD   (* STRIPPER EFFLUENT HOLDING TANK    *)
+--------+

+--------+
|LIC_240|         LANG: LD   (* TREATED EFFLUENT HOLDING TANK     *)
+--------+

+--------+
|LIC_211|         LANG: LD   (* ACID STORAGE TANK                 *)
+--------+
+--------+
|FQ_106 |        LANG: LD  (* SHALLOW WELL COMB FLOW TOTALIZER *)
+--------+

+--------+
|FI_107 |        LANG: LD  (* DEEP WELL COMB FLOW              *)
+--------+

+--------+
|FQ_107 |        LANG: LD  (* DEEP WELL COMB FLOW TOTALIZER    *)
+--------+

+--------+
|FIC_110|        LANG: LD  (* AIR STRIPPER FEED FLOW           *)
+--------+

+--------+
|FIC_220|        LANG: LD  (* GAC ADSORBER FEED FLOW           *)
+--------+

+--------+
|FQ_240 |        LANG: LD  (* TREATED EFFLUENT TOTALIZER       *)
+--------+

+--------+
|LIC_110|        LANG: LD  (* GROUND WATER STORAGE TANK        *)
+--------+

+--------+
|LIC_121|        LANG: LD  (* CAUSTIC STORAGE TANK             *)
+--------+

+--------+
|LIC_131|        LANG: LD  (* CLARIFIER                        *)
+--------+

+--------+
|LIC_145|        LANG: LD  (* HEAD TANK                        *)
+--------+

+--------+
|LIC_220|        LANG: LD  (* STRIPPER EFFLUENT HOLDING TANK   *)
+--------+

+--------+
|LIC_240|        LANG: LD  (* TREATED EFFLUENT HOLDING TANK    *)
+--------+

+--------+
|LIC_211|        LANG: LD  (* ACID STORAGE TANK                *)
+--------+
                 +--------+
                 |LIC_205|        LANG: LD   (* BACKWASH HOLDING TANK                   *)
                 +--------+

|[             INTERRUPTS              ]

                  +------+          +-------------+
                  |TIMER+---------->CALL  FQ_106 |
                  |1.00s|           +-------------+
                  |     |
           CONST -+INTVL|
            00001 |     |
                  |     |
                 -+DELAY|
                  +------+

                  +------+          +-------------+
                  |TIMER+---------->CALL  FQ_107 |
                  |1.00s|           +-------------+
                  |     |
           CONST -+INTVL|
            00001 |     |
                  |     |
                 -+DELAY|
                  +------+

                  +------+          +-------------+
                  |TIMER+---------->CALL  FQ_240 |
                  |1.00s|           +-------------+
                  |     |
           CONST -+INTVL|
            00001 |     |
                  |     |
                 -+DELAY|
                  +------+

|[      START OF PROGRAM LOGIC         ]
|
| << RUNG 5 >>
|
|+-------------+
++CALL  E_STOP +
|+-------------+
|
| << RUNG 6 >>
|
|+-------------+
++CALL  INTLKS +
|+-------------+
|
| << RUNG 7 >>
|
|+-------------+
++CALL   PUMPS +
|+-------------+
|
| << RUNG 8 >>
|
|+-------------+
++CALL   FLOWS +
|+-------------+
|
| << RUNG 9 >>
|
|+-------------+
++CALL   LEVEL +
|+-------------+
|
| << RUNG 10 >>
|
|+-------------+
++CALL    PH   +
|+-------------+
|
| << RUNG 11 >>
|
|+-------------+
++CALL  POLYMER+
|+-------------+
|
| << RUNG 12 >>
|
|+-------------+
++CALL   PRESS +
|+-------------+
|
| << RUNG 13 >>
|
|+-------------+
++CALL   MISC  +
|+-------------+
|
|[      END OF PROGRAM LOGIC      ]
|
+[  START  OF  LD  BLOCK INTLKS    ]
|
|       VARIABLE DECLARATIONS       ]

          V A R I A B L E   D E C L A R A T I O N   T A B L E

       REFERENCE     NICKNAME          REFERENCE DESCRIPTION
       ---------     --------      --------------------------------
                       NO  VARIABLE  TABLE  ENTRIES


                    I D E N T I F I E R   T A B L E

       IDENTIFIER    IDENTIFIER TYPE        IDENTIFIER DESCRIPTION
       ----------    ---------------    --------------------------------
                      NO  IDENTIFIER  TABLE  ENTRIES


+[       START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|LAL110                                                                 I110
|%M00074                                                             %M00110
+--] [-----------------------------------------------------------------( )--
|
|    RUNG 4 >>
|
|AAH130                                                                 I121
|%M00080                                                             %M00121
+--] [-----------------------------------------------------------------( )--
|
| << RUNG 5 >>
|
|LAH121                                                                 I122
|%M00082                                                             %M00122
+--] [-----------------------------------------------------------------( )--
|
| << RUNG 6 >>
|
|LAL121                                                                 I123
|%M00083                                                             %M00123
+--] [-----------------------------------------------------------------( )--
|
| << RUNG 7 >>
|
|LAL145                                                                 I145
|%M00086                                                             %M00145
+--] [-----------------------------------------------------------------( )--
|
| << RUNG 8 >>
|
|    I45                                                                      I146
|%M00085                                                                    %M00146
+--] [--------------------------------------------------------------------------( )--
|
| << RUNG 9 >>
|
|LAHH110                                                                      I150
|%M00072                                                                    %M00150
+--] [--------------------------------------------------------------------------( )--
|
| << RUNG 10 >>
|
|LAH110                                                                       I151
|%M00073                                                                    %M00151
+--] [--------------------------------------------------------------------------( )--
|
| << RUNG 11 >>
|
|PAL150                                                                       I152
|%M00033                                                                    %M00152
+--] [--------------------------------------------------------------------------( )--
|
| << RUNG 12 >>
|
|PAL200                                                                       I200
|    )041                                                                   %M00200
+--] [--------------------------------------------------------------------------( )--
|
| << RUNG 13 >>
|
|AAL200                                                                       I201
|%M00097                                                                    %M00201
+--] [--------------------------------------------------------------------------( )--
|
| << RUNG 14 >>
|
|LAH205                                                                       I205
|%M00032                                                                    %M00205
+--] [--------------------------------------------------------------------------( )--
|
| << RUNG 15 >>
|
|LAL205                                                                       I206
|%M00031                                                                    %M00206
+--] [--------------------------------------------------------------------------( )--
|
| << RUNG 16 >>
|
|P110AS   P110BS                                                              I211
|%I00101  %I00102                                                           %M00211
+   /[------]/[-----------------------------------------------------------------( )--
|
| << RUNG 17 >>
|
|  211                                                                          I212
|%M00092                                                                    %M00212
+--] [--------------------------------------------------------------------------( )--
|
| << RUNG 18 >>
|
|LAH211                                                                         I213
|%M00091                                                                    %M00213
+--] [--------------------------------------------------------------------------( )--
|
| << RUNG 19 >>
|
|LAL220                                                                         I220
|%M00095                                                                    %M00220
+--] [--------------------------------------------------------------------------( )--
|
| << RUNG 20 >>
|
|LAH220                                                                         I221
|%M00094                                                                    %M00221
+--] [--------------------------------------------------------------------------( )--
|
| << RUNG 21 >>
|
|LAHH220                                                                        I222
|    )093                                                                   %M00222
+--] [--------------------------------------------------------------------------( )--
|
| << RUNG 22 >>
|
|LAL240                                                                         I240
|%M00034 +------+                                                           %M00240
+--] [---+ TMR  +---------------------------------------------------------------( )--
|        |1.00s|
|        |      |
| CONST -+PV  CV+-
| +00300 |      |
|        +------+
|        TMR_240
|        %R00302
|
| << RUNG 23 >>
|
|LAH240                                                                         I242
|%M00099                                                                    %M00242
+--] [--------------------------------------------------------------------------( )--
|
+[       END OF BLOCK LOGIC       ]
|
+[  START  OF  LD  BLOCK E_STOP    ]
|
|         VARIABLE DECLARATIONS          ]

              V A R I A B L E   D E C L A R A T I O N   T A B L E

          REFERENCE      NICKNAME          REFERENCE DESCRIPTION
          ---------      --------     --------------------------------
                           NO  VARIABLE  TABLE  ENTRIES


                          I D E N T I F I E R   T A B L E

          IDENTIFIER     IDENTIFIER TYPE          IDENTIFIER DESCRIPTION
          ----------     ---------------     --------------------------------
                          NO  IDENTIFIER  TABLE  ENTRIES


+[        START OF BLOCK LOGIC        ]
|
| << RUNG 3 >>
|
| ESTOP                                                                  ALLSTOP
|%I00124                                                                 %M00001
+--]/[-----------------------------------------------------------------------(S)--
|
|    RUNG 4 >>
|
| ESTOP    RESET                                                         ALLSTOP
|%I00124 %M00002                                                         %M00001
+--] [-----] [---------------------------------------------------------------(R)--
|
+[         END OF BLOCK LOGIC         ]
|
+[  START  OF  LD  BLOCK    PH       ]
|
|          VARIABLE DECLARATIONS        ]

               V A R I A B L E   D E C L A R A T I O N   T A B L E

           REFERENCE     NICKNAME          REFERENCE DESCRIPTION
           ---------     --------     --------------------------------
                           NO  VARIABLE  TABLE  ENTRIES


                          I D E N T I F I E R   T A B L E

           IDENTIFIER     IDENTIFIER TYPE         IDENTIFIER DESCRIPTION
           ----------     ---------------    --------------------------------
                          NO  IDENTIFIER  TABLE  ENTRIES


+[        START OF BLOCK LOGIC        ]
|
| << RUNG 3 >>
|
|         +------+
+---------+ PID_+-
|         | ISA |
|         |     |
|    O_S  |     |  SC121
|%R00210-+SP  CV+-%AQ0005
|         |     |
| AI130   |     |
|%R00211-+PV    |
|         |     |
|       -+MAN   |
|         |     |
|       -+UP    |
|         |     |
|       -+DN    |
|         |     |
|         +------+
|          AC130
|         %R00560
|
| << RUNG 4 >>
|
|          +------+
+----------+ PID_ +-
|          | ISA  |
|          |      |
|A200_S    |      |  SC211
|%R00213--+SP  CV+--%AQ0001
|          |      |
| AI200    |      |
|%R00214--+PV     |
|          |      |
|        --+MAN   |
|          |      |
|        --+UP    |
|          |      |
|        --+DN    |
|          |      |
|          +------+
|           AC200
|          %R00600
|
| << RUNG 5 >>
|
|          +------+
+----------+ EQ_  +-
|          | INT  |
|          |      |
|AI30SEL   |      |         +------+
|%R00300--+I1    Q+---------+MOVE_+-
|          |      |         | INT  |
|          |      |         |      |
|          |      | AIT130A |      |  AI130
| CONST  --+I2    | %AI0018-+IN   Q+--%R00211
| +00001  +------+          | LEN  |
|                           |00001 |
|                           |      |
|                           +------+
|
| << RUNG 6 >>
|
|          +------+
+---------+ EQ_  +-
|          | INT |
|          |      |
|A130SEL  |      |          +------+
|%R00300-+I1   Q+----------+MOVE_+-
|          |      |          | INT  |
|          |      |          |      |
|          |      | AIT130B |      |  AI130
| CONST  -+I2    | %AI0020-+IN   Q+-%R00211
| +00002 +------+           | LEN  |
|                            |00001|
|                            |      |
|                            +------+
|
| << RUNG 7 >>
|
|          +------+
+---------+ EQ_  +-
|          | INT |
|          |      |
|A200SEL  |      |          +------+
|%R00301-+I1   Q+----------+MOVE_+-
|          |      |          | INT  |
|          |      |          |      |
|          |      | AIT200A |      |  AI200
| CONST  -+I2    | %AI0005-+IN   Q+-%R00214
| +00001 +------+           | LEN  |
|                            |00001|
|                            |      |
|                            +------+
|
| << RUNG 8 >>
|
|          +------+
+---------+ EQ_  +-
|          | INT |
|          |      |
|A200SEL  |      |          +------+
|%R00301-+I1   Q+----------+MOVE_+-
|          |      |          | INT  |
|          |      |          |      |
|          |      | AIT200B |      |  AI200
| CONST  -+I2    | %AI0004-+IN   Q+-%R00214
| +00002 +------+           | LEN  |
|                            |00001|
|                            |      |
|                            +------+
|
| << RUNG 9 >>
|
|            +------+
+----------+ EQ_  +-
|          | INT  |
|          |      |
|A200SEL   |      |         +------+
|%R00301--+I1   Q+----------+MOVE_+-
|          |      |         | INT  |
|          |      |         |      |
|          |      | AIT200C |      |  AI200
| CONST  --+I2    | %AI0022-+IN   Q+-%R00214
| +00003  +------+          | LEN  |
|                           |00001|
|                           |      |
|                           +------+
|
+[        END OF BLOCK LOGIC        ]
|
+[  START  OF  LD  BLOCK POLYMER  ]
|
|.      VARIABLE DECLARATIONS      ]

        V A R I A B L E   D E C L A R A T I O N   T A B L E

        REFERENCE     NICKNAME          REFERENCE DESCRIPTION
        ---------     --------     --------------------------------
                         NO  VARIABLE  TABLE  ENTRIES


                     I D E N T I F I E R   T A B L E

        IDENTIFIER     IDENTIFIER TYPE         IDENTIFIER DESCRIPTION
        ----------     ---------------     --------------------------------
                        NO  IDENTIFIER  TABLE  ENTRIES


+[       START OF BLOCK LOGIC      ]
|
|(*  COMMENT  *)
|
| << RUNG 4 >>
|
| I123     I146     I151     X132X                                          X132C
|%I00123 %M00146 %M00151 %M00420                                          %Q00027
|    /[-----]/[-----]/[-----] [--------------------------------------------( )--
|
+[       END OF BLOCK LOGIC       ]
|
+[  START  OF  LD  BLOCK  FLOWS     ]
|
|      VARIABLE DECLARATIONS        ]

          V A R I A B L E   D E C L A R A T I O N   T A B L E

      REFERENCE      NICKNAME           REFERENCE DESCRIPTION
      ---------      --------      --------------------------------
                          NO  VARIABLE  TABLE  ENTRIES


                       I D E N T I F I E R   T A B L E

      IDENTIFIER    IDENTIFIER TYPE         IDENTIFIER DESCRIPTION
      ----------    ---------------    --------------------------------
                         NO  IDENTIFIER  TABLE  ENTRIES


+[      START OF BLOCK LOGIC        ]
|
| << RUNG 3 >>
|
|+-------------+
++CALL  FI_106 +
|+-------------+
|
|    RUNG 4 >>
|
|+-------------+
++CALL  FI_107 +
|+-------------+
|
| << RUNG 5 >>
|
|+-------------+
++CALL  FIC_110+
|+-------------+
|
| << RUNG 6 >>
|
|+-------------+
++CALL  FIC_220+
|+-------------+
|
| << RUNG 7 >>
|
|+-------------+
++CALL  FQ_240 +
|+-------------+
|
| << RUNG 8 >>
|
|[illegible]_EXE
|%S00121          +------+
+--] [-----------+MOVE_+-
|                | DINT|
|                |     |
|                |     | FQ106_D
|         CONST -+IN  Q+-%R00025
|     +0000000000 | LEN |
|                |00001|
|                |     |
|                +------+
|
+[        END OF BLOCK LOGIC        ]
|
+[  START  OF  LD  BLOCK  PRESS    ]
|
|       VARIABLE DECLARATIONS      ]

          V A R I A B L E   D E C L A R A T I O N   T A B L E

        REFERENCE     NICKNAME          REFERENCE DESCRIPTION
        ---------     --------      --------------------------------
                          NO  VARIABLE  TABLE  ENTRIES


                        I D E N T I F I E R   T A B L E

        IDENTIFIER    IDENTIFIER TYPE          IDENTIFIER DESCRIPTION
        ----------    ---------------     --------------------------------
                         NO  IDENTIFIER  TABLE  ENTRIES


+[      START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|         +------+
+---------+ LT_  +-
|         | INT  |
|         |      |                                                  PAL108
|[illegible]08   |      |                                                  %M00079
|%AI0002-+I1   Q+-------------------------------------------------( )--
|         |      |
|P108_L   |      |
|%R00101-+I2     |
|         +------+
|
| << RUNG 4 >>
|
|         +------+
+---------+ GT_  +-
|         | INT  |
|         |      |                                                  PDH220A
|PDT220A  |      |                                                  %M00036
|%AI0013-+I1   Q+-------------------------------------------------( )--
|         |      |
|P220A_H  |      |
|%R00121-+I2     |
|         +------+
|
| << RUNG 5 >>
|
|          +-----+
+----------+ GT_ +-
|          | INT |
|          |     |                                                          PDH220B
|PDT220B   |     |                                                          %M00037
|%AI0010--+I1   Q+---------------------------------------------------------( )--
|          |     |
|P220B_H   |     |
|%R00122--+I2    |
|          +-----+
|
| << RUNG 6 >>
|
|PSL200                                                                     PAL200
|%I00115                                                                    %M00041
+--]/[---------------------------------------------------------------------( )--
|
| << RUNG 7 >>
|
|PSH200                                                                     PAH200
|%I00108                                                                    %M00040
+--] [---------------------------------------------------------------------( )--
|
+[        END OF BLOCK LOGIC        ]
|
+[  START  OF  LD  BLOCK    AIR     ]
|
|       VARIABLE DECLARATIONS       ]

                V A R I A B L E   D E C L A R A T I O N   T A B L E

            REFERENCE      NICKNAME          REFERENCE DESCRIPTION
            ---------      --------      --------------------------------
                              NO  VARIABLE  TABLE  ENTRIES


                          I D E N T I F I E R   T A B L E

            IDENTIFIER     IDENTIFIER TYPE          IDENTIFIER DESCRIPTION
            ----------     ---------------     --------------------------------
                             NO  IDENTIFIER  TABLE  ENTRIES

+[        START OF BLOCK LOGIC      ]
|
| << RUNG 3 >>
|
|PSL152                                                                        PAL150
|%I00087                                                                       %M00033
+--] [----------------------------------------------------------------------------( )--
|
|          END OF BLOCK LOGIC       ]
|
+[  START  OF  LD  BLOCK  LEVEL    ]
|
|      VARIABLE DECLARATIONS       ]

              V A R I A B L E   D E C L A R A T I O N   T A B L E

          REFERENCE     NICKNAME          REFERENCE DESCRIPTION
          ---------     --------    --------------------------------
                           NO  VARIABLE  TABLE  ENTRIES


                           I D E N T I F I E R   T A B L E

          IDENTIFIER    IDENTIFIER TYPE        IDENTIFIER DESCRIPTION
          ----------    ---------------  --------------------------------
                          NO  IDENTIFIER  TABLE  ENTRIES


+[       START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|+-------------+
++CALL  LIC_110+
|+-------------+
|
|    RUNG 4 >>
|
|+-------------+
++CALL  LIC_121+
|+-------------+
|
| << RUNG 5 >>
|
|+-------------+
++CALL  LIC_131+
|+-------------+
|
| << RUNG 6 >>
|
|+-------------+
++CALL  LIC_145+
|+-------------+
|
| << RUNG 7 >>
|
|+-------------+
++CALL  LIC_205+
|+-------------+
|
| << RUNG 8 >>
|
|+-------------+
++CALL  LIC_211+
|+-------------+
|
| << RUNG 9 >>
|
|+-------------+
++CALL  LIC_220+
|+-------------+
|
| << RUNG 10 >>
|
|+-------------+
++CALL  LIC_240+
|+-------------+
|
+[        END OF BLOCK LOGIC        ]
|
+[  START  OF  LD  BLOCK  PUMPS    ]
|
|        VARIABLE DECLARATIONS      ]

          V A R I A B L E   D E C L A R A T I O N   T A B L E

        REFERENCE     NICKNAME           REFERENCE DESCRIPTION
        ---------     --------     --------------------------------
                         NO  VARIABLE  TABLE  ENTRIES


                      I D E N T I F I E R   T A B L E

        IDENTIFIER    IDENTIFIER TYPE         IDENTIFIER DESCRIPTION
        ----------    ---------------    --------------------------------
                        NO  IDENTIFIER  TABLE  ENTRIES


+[       START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|+-------------+
++CALL   P_100 +
|+-------------+
|
|   RUNG 4 >>
|
|+-------------+
++CALL   P_101 +
|+-------------+
|
| << RUNG 5 >>
|
|+-------------+
++CALL   P_102 +
|+-------------+
|
| << RUNG 6 >>
|
|+-------------+
++CALL   P_103 +
|+-------------+
|
| << RUNG 7 >>
|
|+-------------+
++CALL   P_104 +
|+-------------+
|
```

P_025	P_205
~~[illegible]~~	P_211
P_100	P_220A
P_101	P_220B
P_102	P_241
P_103	P_245
P_104	P_300
P_105	P_301
P_106	P_302
P_110A	P_304
P_110B	
P_115	
P_120	
P_121	
P_145	

```
| << RUNG 8 >>
|
|+-------------+
++CALL   P_105 +
|+-------------+
|
| << RUNG 9 >>
|
|+-------------+
++CALL   P_025 +
|+-------------+
|
| << RUNG 10 >>
|
|+-------------+
++CALL   P_115 +
|+-------------+
|
| << RUNG 11 >>
|
|+-------------+
++CALL   P_121 +
|+-------------+
|
| << RUNG 12 >>
|
|+-------------+
++CALL   P_145 +
|+-------------+
|
| << RUNG 13 >>
|
|+-------------+
++CALL   P_211 +
|+-------------+
|
| << RUNG 14 >>
|
|+-------------+
++CALL   P_120 +
|+-------------+
|
| << RUNG 15 >>
|
|+-------------+
++CALL  P_110A +
|+-------------+
|
| << RUNG 16 >>
|
|+-------------+
++CALL  P_110B +
|+-------------+
|
| << RUNG 17 >>
|
|  ------------+
++CALL   P_205 +
|+-------------+
|
| << RUNG 18 >>
|
|+-------------+
++CALL  P_220A +
|+-------------+
|
| << RUNG 19 >>
|
|+-------------+
++CALL  P_220B +
|+-------------+
|
| << RUNG 20 >>
|
|+-------------+
++CALL   P_241 +
|+-------------+
|
| << RUNG 21 >>
|
|+-------------+
+   LL   P_245 +
|+-------------+
|
| << RUNG 22 >>
|
|+-------------+
++CALL   P_300 +
|+-------------+
|
| << RUNG 23 >>
|
|+-------------+
++CALL   P_301 +
|+-------------+
|
| << RUNG 24 >>
|
|+-------------+
++CALL   P_302 +
|+-------------+
|
| << RUNG 25 >>
|
|+-------------+
++CALL   P_304 +
|  -----------+
|
+[      END OF BLOCK LOGIC         ]
|
+[  START  OF  LD  BLOCK  MISC      ]
|
|       VARIABLE DECLARATIONS       ]

         V A R I A B L E   D E C L A R A T I O N   T A B L E

     REFERENCE      NICKNAME          REFERENCE DESCRIPTION
     ---------      --------      --------------------------------
                        NO  VARIABLE  TABLE  ENTRIES


                   I D E N T I F I E R   T A B L E

     IDENTIFIER     IDENTIFIER TYPE         IDENTIFIER DESCRIPTION
     ----------     ---------------    --------------------------------
                       NO  IDENTIFIER  TABLE  ENTRIES

+[       START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|+-------------+
++CALL    AIR  +
|+-------------+
|
|   . RUNG 4 >>
|
|+-------------+
++CALL   K_200 +
|+-------------+
|
+[        END OF BLOCK LOGIC        ]
|
+[  START  OF  LD  BLOCK  P_100     ]
|
|      VARIABLE DECLARATIONS        ]

          V A R I A B L E   D E C L A R A T I O N   T A B L E

          REFERENCE     NICKNAME         REFERENCE DESCRIPTION
          ---------     --------      --------------------------------
                           NO  VARIABLE  TABLE  ENTRIES


                          I D E N T I F I E R   T A B L E

          IDENTIFIER    IDENTIFIER TYPE        IDENTIFIER DESCRIPTION
          ----------    ---------------     --------------------------------
                          NO  IDENTIFIER  TABLE  ENTRIES


+[       START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|LSL100                                                                  LAL100
|%I00128                                                                %M00043
+--]/[----------------------------------------------------------------------( )--
|
|    RUNG 4 >>
|
|LAL100                                                                   I100
|%M00043                                                                %M00100
+--] [----------------------------------------------------------------------( )--
|
| << RUNG 5 >>
|
| P100I   P100D                                                           P100C
|%M00313 %M00314                                                        %Q00022
+--] [------]/[-------------------------------------------------------------( )--
|
| << RUNG 6 >>
|
| P100X   I150                                                            P100I
|%M00308 %M00150                                                        %M00313
+--] [------]/[-------------------------------------------------------------( )--
|
| << RUNG 7 >>
|
|   23                                                                          P100D
|%M00123                                                                      %M00314
+--] [--+-----------------------------------------------------------------------(S)--
|       |
| I146  |
|%M00146|
+--] [--+
|       |
| I151  |
|%M00151|
+--] [--+
|       |
|ALLSTOP|
|%M00001|
+--] [--+
|
| << RUNG 8 >>
|
| I123    I146    I151   ALLSTOP  RESET                                         P100D
|%M00123 %M00146 %M00151 %M00001 %M00002                                      %M00314
+--]/[-----]/[-----]/[-----]/[-----] [------------------------------------------(R)--
|
+[       END OF BLOCK LOGIC       ]
|
+[  START  OF  LD  BLOCK  P_101    ]
|
|       VARIABLE DECLARATIONS       ]

            V A R I A B L E   D E C L A R A T I O N   T A B L E

         REFERENCE      NICKNAME          REFERENCE DESCRIPTION
         ---------      --------      --------------------------------
                           NO  VARIABLE  TABLE  ENTRIES


                           I D E N T I F I E R   T A B L E

         IDENTIFIER    IDENTIFIER TYPE        IDENTIFIER DESCRIPTION
         ----------    ---------------    ----------------------------------
                          NO  IDENTIFIER  TABLE  ENTRIES


+[       START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|LSL101                                                                 LAL101
|%I00093                                                               %M00045
+--]/[--------------------------------------------------------------------( )--
|
|    RUNG 4 >>
|
|LAL101                                                                   I101
|%M00045                                                               %M00101
+--] [--------------------------------------------------------------------( )--
|
| << RUNG 5 >>
|
| P101I    P101D                                                         P101C
|%M00320 %M00321                                                       %Q00003
+--] [------]/[-----------------------------------------------------------( )--
|
| << RUNG 6 >>
|
| P101X                                                                  P101I
|%M00315                                                               %M00320
+--] [--------------------------------------------------------------------( )--
|
| << RUNG 7 >>
|
|  .50                                                                     P101D
|%M00150                                                                  %M00321
+--] [--+------------------------------------------------------------------(S)--
|       |
| I151  |
|%M00151|
+--] [--+
|       |
| I212  |
|%M00212|
+--] [--+
|       |
|ALLSTOP|
|%M00001|
+--] [--+
|
| << RUNG 8 >>
|
| I150    I151    I212   ALLSTOP  RESET                                    P101D
|%M00150 %M00151 %M00212 %M00001 %M00002                                  %M00321
+--]/[-----]/[-----]/[-----]/[-----] [-------------------------------------(R)--
|
+[       END OF BLOCK LOGIC       ]
|
+[    START  OF  LD  BLOCK  P_102    ]
|
|        VARIABLE DECLARATIONS       ]

              V A R I A B L E   D E C L A R A T I O N   T A B L E

          REFERENCE     NICKNAME          REFERENCE DESCRIPTION
          ---------     --------      --------------------------------
                           NO  VARIABLE  TABLE  ENTRIES


                         I D E N T I F I E R   T A B L E

          IDENTIFIER     IDENTIFIER TYPE          IDENTIFIER DESCRIPTION
          ----------     ---------------     --------------------------------
                          NO  IDENTIFIER  TABLE  ENTRIES

+[        START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|LSL102                                                                    LAL102
|%I00130                                                                  %M00047
+--]/[--------------------------------------------------------------------( )--
|
| << RUNG 4 >>
|
|LAL102                                                                      I102
|%M00047                                                                  %M00102
+--] [--------------------------------------------------------------------( )--
|
| << RUNG 5 >>
|
| P102I    P102D                                                            P102C
|%M00327 %M00328                                                          %Q00023
+--] [------]/[-----------------------------------------------------------( )--
|
| << RUNG 6 >>
|
| P102X    I150                                                             P102I
|%M00322 %M00150                                                          %M00327
+--] [------]/[-----------------------------------------------------------( )--
|
| << RUNG 7 >>
|
|   23                                                                        P102D
|%M00123                                                                     %M00328
+--] [--+---------------------------------------------------------------------(S)--
|       |
| I146  |
|%M00146|
+--] [--+
|       |
| I151  |
|%M00151|
+--] [--+
|       |
|ALLSTOP|
|%M00001|
+--] [--+
|
| << RUNG 8 >>
|
| I123    I146    I151   ALLSTOP  RESET                                        P102D
|%M00123 %M00146 %M00151 %M00001 %M00002                                      %M00328
+--]/[-----]/[-----]/[-----]/[-----] [-----------------------------------------(R)--
|
+[       END OF BLOCK LOGIC        ]
|
+[  START  OF  LD  BLOCK  P_103    ]
|
|      VARIABLE DECLARATIONS       ]

                V A R I A B L E   D E C L A R A T I O N   T A B L E

           REFERENCE     NICKNAME          REFERENCE DESCRIPTION
           ---------     --------     --------------------------------
                              NO  VARIABLE  TABLE  ENTRIES


                           I D E N T I F I E R   T A B L E

           IDENTIFIER    IDENTIFIER TYPE        IDENTIFIER DESCRIPTION
           ----------    ---------------    --------------------------------
                             NO  IDENTIFIER  TABLE  ENTRIES

+[       START OF BLOCK LOGIC      ]
|
| << RUNG 3 >>
|
|LSL103                                                                    LAL103
|%I00097                                                                  %M00049
+--]/[-------------------------------------------------------------------( )--
|
|   RUNG 4 >>
|
|LAL103                                                                     I103
|%M00049                                                                  %M00103
+--] [-------------------------------------------------------------------( )--
|
| << RUNG 5 >>
|
| P103I   P103D                                                            P103C
|%M00334 %M00335                                                          %Q00004
+--] [-----]/[-----------------------------------------------------------( )--
|
| << RUNG 6 >>
|
| P103X                                                                    P103I
|%M00329                                                                  %M00334
+--] [-------------------------------------------------------------------( )--
|
| << RUNG 7 >>
|
|  I150                                                                                    P103D
|%M00150                                                                                 %M00335
+--] [--+-----------------------------------------------------------------------------------(S)--
|       |
| I151  |
|%M00151|
+--] [--+
|       |
| I212  |
|%M00212|
+--] [--+
|       |
|ALLSTOP|
|%M00001|
+--] [--+
|
| << RUNG 8 >>
|
| I150     I151     I212    ALLSTOP  RESET                                                 P103D
|%M00150 %M00151 %M00212 %M00001 %M00002                                                  %M00335
+--]/[-----]/[-----]/[-----]/[-----] [---------------------------------------------------------(R)--
|
+[        END OF BLOCK LOGIC        ]
|
+[  START  OF  LD  BLOCK  P_104    ]

|      VARIABLE DECLARATIONS       ]

         V A R I A B L E   D E C L A R A T I O N   T A B L E

        REFERENCE     NICKNAME         REFERENCE DESCRIPTION
        ---------     --------     --------------------------------
                         NO  VARIABLE  TABLE  ENTRIES


                       I D E N T I F I E R   T A B L E

        IDENTIFIER     IDENTIFIER TYPE        IDENTIFIER DESCRIPTION
        ----------     ---------------     --------------------------------
                        NO  IDENTIFIER  TABLE  ENTRIES


+[      START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|LSL104                                                                    LAL104
|%I00132                                                                   %M00051
+--]/[-----------------------------------------------------------------------( )--
|
|     RUNG 4 >>
|
|LAL104                                                                     I104
|%M00051                                                                   %M00104
+--] [-----------------------------------------------------------------------( )--
|
| << RUNG 5 >>
|
| P104I   P104D                                                             P104C
|%M00341 %M00342                                                           %Q00038
+--] [-----]/[---------------------------------------------------------------( )--
|
| << RUNG 6 >>
|
| P104X   I150                                                              P104I
|%M00336 %M00150                                                           %M00341
+--] [-----]/[---------------------------------------------------------------( )--
|
| << RUNG 7 >>
|
| I123                                                                        P104D
|%M00123                                                                     %M00342
+--] [--+-------------------------------------------------------------------------(S)--
|       |
| I146  |
|%M00146|
+--] [--+
|       |
| I151  |
|%M00151|
+--] [--+
|       |
|ALLSTOP|
|%M00001|
+--] [--+
|
| << RUNG 8 >>
|
| I123     I146     I151    ALLSTOP   RESET                                        P104D
|%M00123 %M00146 %M00151 %M00001 %M00002                                          %M00342
+--]/[-----]/[-----]/[-----]/[-----] [--------------------------------------------(R)--
|
+[       END OF BLOCK LOGIC       ]
|
+[  START  OF  LD  BLOCK  P_105    ]

         VARIABLE DECLARATIONS          ]

              V A R I A B L E   D E C L A R A T I O N   T A B L E

          REFERENCE      NICKNAME         REFERENCE DESCRIPTION
          ---------      --------     --------------------------------
                            NO  VARIABLE  TABLE  ENTRIES


                         I D E N T I F I E R   T A B L E

          IDENTIFIER     IDENTIFIER TYPE         IDENTIFIER DESCRIPTION
          ----------     ---------------     --------------------------------
                           NO  IDENTIFIER  TABLE  ENTRIES


+[      START OF BLOCK LOGIC        ]
|
| << RUNG 3 >>
|
|LSL105                                                                   LAL105
|%I00100                                                                 %M00053
+--]/[-------------------------------------------------------------------( )--
|
|  RUNG 4 >>
|
|LAL105                                                                    I105
|%M00053                                                                 %M00105
+--] [-------------------------------------------------------------------( )--
|
| << RUNG 5 >>
|
| P105I   P105D                                                            P105C
|%M00348 %M00349                                                         %Q00005
+--] [-----]/[-----------------------------------------------------------( )--
|
| << RUNG 6 >>
|
| P105X                                                                    P105I
|%M00343                                                                 %M00348
+--] [-------------------------------------------------------------------( )--
|
| << RUNG 7 >>
|
|    50                                                                          P105D
|%M00150                                                                       %M00349
+--] [--+----------------------------------------------------------------------------(S)--
|       |
| I151  |
|%M00151|
+--] [--+
|       |
| I212  |
|%M00212|
+--] [--+
|       |
|ALLSTOP|
|%M00001|
+--] [--+
|
| << RUNG 8 >>
|
| I150    I151    I212   ALLSTOP  RESET                                          P105D
|%M00150 %M00151 %M00212 %M00001 %M00002                                       %M00349
+--]/[-----]/[-----]/[-----]/[-----] [-------------------------------------------(R)--
|
+[       END OF BLOCK LOGIC       ]
|
+[  START  OF  LD  BLOCK  P_025    ]
|
|       VARIABLE DECLARATIONS        ]

                 V A R I A B L E   D E C L A R A T I O N   T A B L E

           REFERENCE      NICKNAME          REFERENCE DESCRIPTION
           ---------      --------      --------------------------------
                           NO  VARIABLE  TABLE  ENTRIES


                           I D E N T I F I E R   T A B L E

           IDENTIFIER     IDENTIFIER TYPE       IDENTIFIER DESCRIPTION
           ----------     ---------------    ---------------------------------
                          NO  IDENTIFIER  TABLE  ENTRIES

+[       START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|LSA025                                                                    LAH025
|%I00104                                                                   %M00042
+--] [-----------------------------------------------------------------------( )--
|
|  << RUNG 4 >>
|
|LSH025                                                                     P025C
|%I00118                                                                   %Q00009
+--] [-----------------------------------------------------------------------(S)--
|
| << RUNG 5 >>
|
|LSL025                                                                     P025C
|%I00117                                                                   %Q00009
+--]/[-----------------------------------------------------------------------(R)--
|
+[        END OF BLOCK LOGIC        ]
|
+[  START  OF  LD  BLOCK  P_115    ]
|
      VARIABLE DECLARATIONS        ]

          V A R I A B L E   D E C L A R A T I O N   T A B L E

     REFERENCE    NICKNAME         REFERENCE DESCRIPTION
     ---------    --------     --------------------------------
                   NO  VARIABLE  TABLE  ENTRIES


                   I D E N T I F I E R   T A B L E

     IDENTIFIER    IDENTIFIER TYPE        IDENTIFIER DESCRIPTION
     ----------    ---------------     --------------------------------
                   NO  IDENTIFIER  TABLE  ENTRIES

+[       START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|LSA115                                                              LAH115
|%I00103                                                            %M00077
+--] [--------------------------------------------------------------( )--
|
  : RUNG 4 >>
|
|LSH115                                                               P115C
|%I00088                                                            %Q00008
+--] [--------------------------------------------------------------(S)--
|
| << RUNG 5 >>
|
|LSL115                                                               P115C
|%I00116                                                            %Q00008
+--]/[--------------------------------------------------------------(R)--
|
+[        END OF BLOCK LOGIC        ]
|
+[  START  OF  LD  BLOCK  P_121    ]
!
!      VARIABLE DECLARATIONS       ]

              V A R I A B L E   D E C L A R A T I O N   T A B L E

          REFERENCE     NICKNAME          REFERENCE DESCRIPTION
          ---------     --------     --------------------------------
                             NO  VARIABLE  TABLE  ENTRIES


                          I D E N T I F I E R   T A B L E

          IDENTIFIER     IDENTIFIER TYPE          IDENTIFIER DESCRIPTION
          ----------     ---------------     --------------------------------
                            NO  IDENTIFIER  TABLE  ENTRIES

+[       START OF BLOCK LOGIC       ]
!
! << RUNG 3 >>
!
! P121I   P121D                                                              P121C
!%M00411 %M00412                                                           %Q00002
+--] [-----]/[---------------------------------------------------------------( )--
!
!    RUNG 4 >>
!
! P121X                                                                      P121I
!%M00406                                                                   %M00411
+--] [-----------------------------------------------------------------------( )--
!
! << RUNG 5 >>
!
! I121                                                                       P121D
!%M00121                                                                   %M00412
+--] [--+--------------------------------------------------------------------( )--
!       !
! I123  !
!%M00123!
+--] [--+
!       !
! I146  !
!%M00146!
+--] [--+
!       !
!ALLSTOP!
!%M00001!
+--] [--+
!
+[        END OF BLOCK LOGIC        ]
!
```

```
+[  START  OF  LD  BLOCK  P_145     ]
|
       VARIABLE DECLARATIONS        ]

             V A R I A B L E   D E C L A R A T I O N   T A B L E

          REFERENCE      NICKNAME           REFERENCE DESCRIPTION
          ---------      --------      --------------------------------
                           NO  VARIABLE  TABLE  ENTRIES


                     I D E N T I F I E R   T A B L E

          IDENTIFIER      IDENTIFIER TYPE          IDENTIFIER DESCRIPTION
          ----------      ---------------     --------------------------------
                          NO  IDENTIFIER  TABLE  ENTRIES


+[      START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
| P145I   P145D                                                          P145C
|%M00432 %M00433                                                        %Q00037
+--] [-----]/[----------------------------------------------------------( )--
|
|    RUNG 4 >>
|
| P145X  LCM145                                                          P145I
|%M00427 %M00027                                                        %M00432
+--] [-----] [----------------------------------------------------------( )--
|
| << RUNG 5 >>
|
| I145                                                                   P145D
|%M00145                                                                %M00433
+--] [--+---------------------------------------------------------------( )--
|       |
| I151  |
|%M00151|
+--] [--+
|       |
| I212  |
|%M00212|
+--] [--+
|       |
| I240  |
|%M00240|
+--] [--+
|       |
|ALLSTOP|
|    0001|
+    ] [--+
+[       END OF BLOCK LOGIC        ]
|
+[  START  OF  LD  BLOCK  P_211    ]
|
|      VARIABLE DECLARATIONS       ]

           V A R I A B L E   D E C L A R A T I O N   T A B L E

         REFERENCE     NICKNAME          REFERENCE DESCRIPTION
         ---------     --------     --------------------------------
                          NO  VARIABLE  TABLE  ENTRIES


                      I D E N T I F I E R   T A B L E

         IDENTIFIER     IDENTIFIER TYPE        IDENTIFIER DESCRIPTION
         ----------     ---------------    ---------------------------------
                         NO  IDENTIFIER  TABLE  ENTRIES

+[       START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
| P211I   P211D                                                          P211C
|%M00453 %M00454                                                        %Q00001
+--] [------]/[-------------------------------------------------------------( )--
|
|    RUNG 4 >>
|
| P211X                                                                  P211I
|%M00448                                                                %M00453
+--] [----------------------------------------------------------------------( )--
|
| << RUNG 5 >>
|
   10                                                                     P211D
|%M00110                                                                 %M00454
+--] [--+-------------------------------------------------------------------( )--
|       |
| I150  |
|%M00150|
+--] [--+
|       |
| I201  |
|%M00201|
+--] [--+
|       |
| I211  |
|%M00211|
+--] [--+
|       |
| I212  |
|%M00212|
+--] [--+
|       |
|ALLSTOP|
|%M00001|
+--] [--+
|
+[       END OF BLOCK LOGIC        ]
|
+[  START  OF  LD  BLOCK  P_120   ]
|
|      VARIABLE DECLARATIONS      ]

         V A R I A B L E   D E C L A R A T I O N   T A B L E

          REFERENCE     NICKNAME        REFERENCE DESCRIPTION
          ---------     --------    --------------------------------
                           NO  VARIABLE  TABLE  ENTRIES


                     I D E N T I F I E R   T A B L E

          IDENTIFIER    IDENTIFIER TYPE        IDENTIFIER DESCRIPTION
          ----------    ---------------   --------------------------------
                          NO  IDENTIFIER  TABLE  ENTRIES


+[      START OF BLOCK LOGIC      ]
|
| << RUNG 3 >>
|
| P120X   I110   ALLSTOP                                              P120C
|%M00399 %M00110 %M00001                                             %Q00025
+--] [-----]/[-----]/[---------------------------------------------------( )--
|
+       END OF BLOCK LOGIC      ]
|
+[  START  OF  LD  BLOCK P_110A    ]
|
       VARIABLE DECLARATIONS        ]

           V A R I A B L E   D E C L A R A T I O N   T A B L E

        REFERENCE      NICKNAME          REFERENCE DESCRIPTION
        ---------      --------      --------------------------------
                        NO  VARIABLE  TABLE  ENTRIES


                       I D E N T I F I E R   T A B L E

        IDENTIFIER     IDENTIFIER TYPE         IDENTIFIER DESCRIPTION
        ----------     ---------------     ---------------------------------
                       NO  IDENTIFIER  TABLE  ENTRIES


+[       START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|P110AI  P110AD                                                           P110AC
|%M00383 %M00384                                                          %Q00006
+--] [-----]/[----------------------------------------------------------( )--
|
   RUNG 4 >>
|
|P110AX                                                                   P110AI
|%M00378                                                                  %M00383
+--] [------------------------------------------------------------------( )--
|
| << RUNG 5 >>
|
| I110                                                                    P110AD
|%M00110                                                                  %M00384
+--] [--+---------------------------------------------------------------( )--
|       |
| I200  |
|%M00200|
+--] [--+
|       |
| I221  |
|%M00221|
+--] [--+
|       |
| I240  |
|%M00240|
+--] [--+
|       |
|ALLSTOP|
|   0001|
+   ] [--+
+[      END OF BLOCK LOGIC       ]
+[  START  OF  LD  BLOCK P_110B    ]
|
+[      VARIABLE DECLARATIONS       ]

             V A R I A B L E   D E C L A R A T I O N   T A B L E

         REFERENCE     NICKNAME           REFERENCE DESCRIPTION
         ---------     --------     --------------------------------
                             NO  VARIABLE  TABLE  ENTRIES


                         I D E N T I F I E R   T A B L E

         IDENTIFIER     IDENTIFIER TYPE        IDENTIFIER DESCRIPTION
         ----------     ---------------    --------------------------------
                            NO  IDENTIFIER  TABLE  ENTRIES


+[       START OF BLOCK LOGIC      ]
|
| << RUNG 3 >>
|
|P110BI  P110BD                                                           P110BC
|%M00390 %M00391                                                          %Q00007
+--] [-----]/[------------------------------------------------------------( )--
|
| << RUNG 4 >>
|
|P110BX                                                                   P110BI
|%M00385                                                                  %M00390
+--] [--------------------------------------------------------------------( )--
|
| << RUNG 5 >>
|
| I110                                                                    P110BD
|%M00110                                                                  %M00391
+--] [--+-----------------------------------------------------------------( )--
|       |
| I200  |
|%M00200|
+--] [--+
|       |
| I221  |
|%M00221|
+--] [--+
|       |
| I240  |
|%M00240|
+--] [--+
|       |
|ALLSTOP|
|%M00001|
+--] [--+
+[       END OF BLOCK LOGIC        ]
|
+[  START  OF  LD  BLOCK  P_205    ]
|
|       VARIABLE DECLARATIONS       ]

            V A R I A B L E   D E C L A R A T I O N   T A B L E

          REFERENCE     NICKNAME         REFERENCE DESCRIPTION
          ---------     --------     --------------------------------
                          NO  VARIABLE  TABLE  ENTRIES


                        I D E N T I F I E R   T A B L E

          IDENTIFIER    IDENTIFIER TYPE        IDENTIFIER DESCRIPTION
          ----------    ---------------   ----------------------------------
                         NO  IDENTIFIER  TABLE  ENTRIES


+[      START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
| P205I   P205D                                                           P205C
|%M00446 %M00447                                                         %Q00021
+--] [------]/[---------------------------------------------------------------( )--
|
|     RUNG 4 >>
|
| P205X                                                                   P205I
|%M00441                                                                 %M00446
+--] [------------------------------------------------------------------------( )--
|
| << RUNG 5 >>
|
| I130                                                                    P205D
|%M00130                                                                 %M00447
+--] [--+---------------------------------------------------------------------( )--
|       |
| I206  |
|%M00206|
+--] [--+
|       |
| I240  |
|%M00240|
+--] [--+
|       |
|ALLSTOP|
|%M00001|
+--] [--+
|
+[       END OF BLOCK LOGIC       ]
|
+[  START  OF  LD  BLOCK P_220A     ]
|
        VARIABLE DECLARATIONS        ]

            V A R I A B L E   D E C L A R A T I O N   T A B L E

        REFERENCE      NICKNAME          REFERENCE DESCRIPTION
        ---------      --------     --------------------------------
                          NO  VARIABLE  TABLE  ENTRIES


                         I D E N T I F I E R   T A B L E

        IDENTIFIER     IDENTIFIER TYPE        IDENTIFIER DESCRIPTION
        ----------     ---------------     --------------------------------
                         NO  IDENTIFIER  TABLE  ENTRIES


+[       START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|P220AI  P220AD                                                          P220AC
|%M00460 %M00461                                                         %Q00017
+--] [------]/[---------------------------------------------------------( )--
|
   RUNG 4 >>
|
|P220AX                                                                  P220AI
|%M00455                                                                 %M00460
+--] [------------------------------------------------------------------( )--
|
| << RUNG 5 >>
|
| I220                                                                   P220AD
|%M00220                                                                 %M00461
+--] [--+---------------------------------------------------------------( )--
|       |
| I242  |
|%M00242|
+--] [--+
|       |
|ALLSTOP|
|%M00001|
+--] [--+
|
+[        END OF BLOCK LOGIC        ]
|
+[  START  OF  LD  BLOCK P_220B    ]
:
:       VARIABLE DECLARATIONS       ]

             V A R I A B L E   D E C L A R A T I O N   T A B L E

          REFERENCE     NICKNAME        REFERENCE DESCRIPTION
          ---------     --------    --------------------------------
                           NO  VARIABLE  TABLE  ENTRIES


                         I D E N T I F I E R   T A B L E

          IDENTIFIER     IDENTIFIER TYPE       IDENTIFIER DESCRIPTION
          ----------     ---------------   --------------------------------
                          NO  IDENTIFIER  TABLE  ENTRIES


+[       START OF BLOCK LOGIC       ]
:
: << RUNG 3 >>
:
:P220BI  P220BD                                                      P220BC
:%M00467 %M00468                                                     %Q00018
+--] [-----]/[-------------------------------------------------------( )--
:
:    RUNG 4 >>
:
:P220BX                                                              P220BI
:%M00462                                                             %M00467
+--] [---------------------------------------------------------------( )--
:
: << RUNG 5 >>
:
: I220                                                               P220BD
:%M00220                                                             %M00468
+--] [--+------------------------------------------------------------( )--
:       :
: I242  :
:%M00242:
+--] [--+
:       :
:ALLSTOP:
:%M00001:
+--] [--+
:
+[        END OF BLOCK LOGIC        ]
:
```

```
-[  START  OF  LD  BLOCK  P_245    ]

         VARIABLE DECLARATIONS        ]

              V A R I A B L E   D E C L A R A T I O N   T A B L E

          REFERENCE      NICKNAME          REFERENCE DESCRIPTION
          ---------      --------      ---------------------------------
                           NO  VARIABLE  TABLE  ENTRIES


                         I D E N T I F I E R   T A B L E

          IDENTIFIER     IDENTIFIER TYPE        IDENTIFIER DESCRIPTION
          ----------     ---------------   ----------------------------------
                          NO  IDENTIFIER  TABLE  ENTRIES

-[       START OF BLOCK LOGIC       ]

 << RUNG 3 >>

 P245I   P245D                                                              P245C
%M00481 %M00482                                                            %Q00026
---] [------]/[-------------------------------------------------------------( )--

      RUNG 4 >>

 P245X                                                                      P245I
%M00476                                                                    %M00481
---] [----------------------------------------------------------------------( )--

 << RUNG 5 >>

 I240                                                                       P245D
%M00240                                                                    %M00482
---] [--+-------------------------------------------------------------------( )--
        |
ALLSTOP|
%M00001|
+--] [--+

-[        END OF BLOCK LOGIC        ]
```

```
+[  START  OF  LD  BLOCK  P_241     ]
|
|        VARIABLE DECLARATIONS      ]

            V A R I A B L E   D E C L A R A T I O N   T A B L E

          REFERENCE     NICKNAME          REFERENCE DESCRIPTION
          ---------     --------     --------------------------------
                           NO  VARIABLE  TABLE  ENTRIES


                          I D E N T I F I E R   T A B L E

          IDENTIFIER     IDENTIFIER TYPE        IDENTIFIER DESCRIPTION
          ----------     ---------------     --------------------------------
                          NO  IDENTIFIER  TABLE  ENTRIES

+[        START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
| P241I   P241D                                                         P241C
|%M00474 %M00475                                                      %Q00019
+--] [------]/[----------------------------------------------------------( )--
|
|     RUNG 4 >>
|
| P241X                                                                 P241I
|%M00469                                                              %M00474
+--] [-------------------------------------------------------------------( )--
|
| << RUNG 5 >>
|
| I205                                                                  P241D
|%M00205                                                              %M00475
+--] [--+----------------------------------------------------------------( )--
|       |
| I240  |
|%M00240|
+--] [--+
|       |
|ALLSTOP|
|%M00001|
+--] [--+
|
+[        END OF BLOCK LOGIC         ]
|
+[  START  OF  LD  BLOCK  P_300    ]
|
|      VARIABLE DECLARATIONS       ]

            V A R I A B L E   D E C L A R A T I O N   T A B L E

          REFERENCE     NICKNAME         REFERENCE DESCRIPTION
          ---------     --------     ------------------------------
                            NO  VARIABLE  TABLE  ENTRIES


                          I D E N T I F I E R   T A B L E

          IDENTIFIER     IDENTIFIER TYPE        IDENTIFIER DESCRIPTION
          ----------     ---------------     ------------------------------
                           NO  IDENTIFIER  TABLE  ENTRIES


+[       START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|LSL300                                                              LAL300
|%I00129                                                            %M00055
+--]/[------------------------------------------------------------( )--
|
|    RUNG 4 >>
|
|LAL300                                                                I300
|%M00055                                                            %M00106
+--] [------------------------------------------------------------( )--
|
| << RUNG 5 >>
|
| P300I    P300D                                                      P300C
|%M00355 %M00356                                                    %Q00010
+--] [-----]/[----------------------------------------------------( )--
|
| << RUNG 6 >>
|
| P300X    I150                                                       P300I
|%M00350 %M00150                                                    %M00355
+--] [-----]/[----------------------------------------------------( )--
|
| << RUNG 7 >>
|
|    .23                                                                        P300D
|%M00123                                                                       %M00356
+--] [--+-------------------------------------------------------------------------(S)--
|       |
| I146  |
|%M00146|
+--] [--+
|       |
| I151  |
|%M00151|
+--] [--+
|       |
|ALLSTOP|
|%M00001|
+--] [--+
|
| << RUNG 8 >>
|
| I123     I146     I151    ALLSTOP   RESET                                       P300D
|%M00123 %M00146 %M00151 %M00001 %M00002                                        %M00356
+--]/[-----]/[-----]/[-----]/[-----] [------------------------------------------(R)--
|
+[        END OF BLOCK LOGIC        ]
|
+[  START  OF  LD  BLOCK  P_301    ]
|
|       VARIABLE DECLARATIONS       ]

              V A R I A B L E   D E C L A R A T I O N   T A B L E

          REFERENCE     NICKNAME         REFERENCE DESCRIPTION
          ---------     --------     --------------------------------
                            NO  VARIABLE  TABLE  ENTRIES


                           I D E N T I F I E R    T A B L E

          IDENTIFIER     IDENTIFIER TYPE        IDENTIFIER DESCRIPTION
          ----------     ---------------    --------------------------------
                           NO  IDENTIFIER  TABLE  ENTRIES


+[       START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|LSL301                                                                  LAL301
|%I00138                                                                 %M00061
+--]/[-------------------------------------------------------------------( )--
|
|   RUNG 4 >>
|
|LAL301                                                                   I304
|%M00061                                                                 %M00109
+--] [-------------------------------------------------------------------( )--
|
| << RUNG 5 >>
|
| P301I   P301D                                                           P301C
|%M00376 %M00377                                                         %Q00039
+--] [-----]/[-----------------------------------------------------------( )--
|
| << RUNG 6 >>
|
| P301X                                                                   P301I
|%M00371                                                                 %M00376
+--] [-------------------------------------------------------------------( )--
|
| << RUNG 7 >>
|
|   .23                                                                         P302D
|%M00123                                                                       %M00363
+--] [--+-----------------------------------------------------------------------(S)--
|       |
| I146  |
|%M00146|
+--] [--+
|       |
| I151  |
|%M00151|
+--] [--+
|       |
|ALLSTOP|
|%M00001|
+--] [--+
|
| << RUNG 8 >>
|
| I123    I146    I151    ALLSTOP  RESET                                        P302D
|%M00123 %M00146 %M00151 %M00001 %M00002                                       %M00363
+--]/[-----]/[-----]/[-----]/[-----] [------------------------------------------(R)--
|
+[        END OF BLOCK LOGIC       ]
|
+[  START  OF  LD  BLOCK  P_304    ]
|
|       VARIABLE DECLARATIONS       ]

              V A R I A B L E   D E C L A R A T I O N   T A B L E

          REFERENCE     NICKNAME         REFERENCE DESCRIPTION
          ---------     --------    --------------------------------
                             NO  VARIABLE  TABLE  ENTRIES


                        I D E N T I F I E R   T A B L E

          IDENTIFIER     IDENTIFIER TYPE          IDENTIFIER DESCRIPTION
          ----------     ---------------     --------------------------------
                            NO  IDENTIFIER  TABLE  ENTRIES


+[       START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|LSL304                                                                   LAL304
|%I00133                                                                  %M00059
+--]/[------------------------------------------------------------------( )--
|
|      RUNG 4 >>
|
|LAL304                                                                    I302
|%M00059                                                                  %M00108
+--] [------------------------------------------------------------------( )--
|
| << RUNG 5 >>
|
| P304I   P304D                                                           P304C
|%M00369 %M00370                                                          %Q00012
+--] [-----]/[----------------------------------------------------------( )--
|
| << RUNG 6 >>
|
| P304X    I150                                                           P304I
|%M00364 %M00150                                                          %M00369
+--] [-----]/[----------------------------------------------------------( )--
|
| << RUNG 7 >>
|
|   23                                                                         P304D
|%M00123                                                                      %M00370
+--] [--+-----------------------------------------------------------------------(S)--
|       |
| I146  |
|%M00146|
+--] [--+
|       |
| I151  |
|%M00151|
+--] [--+
|       |
|ALLSTOP|
|%M00001|
+--] [--+
|
| << RUNG 8 >>
|
| I123    I146    I151    ALLSTOP  RESET                                        P304D
|%M00123 %M00146 %M00151 %M00001 %M00002                                       %M00370
+--]/[-----]/[-----]/[-----]/[-----] [------------------------------------------(R)--
|
+[        END OF BLOCK LOGIC         ]
|
+[  START  OF  LD  BLOCK  K_200    ]
|
|      VARIABLE DECLARATIONS       ]

              V A R I A B L E   D E C L A R A T I O N   T A B L E

          REFERENCE      NICKNAME          REFERENCE DESCRIPTION
          ---------      --------      --------------------------------
                             NO  VARIABLE  TABLE  ENTRIES


                          I D E N T I F I E R   T A B L E

          IDENTIFIER     IDENTIFIER TYPE          IDENTIFIER DESCRIPTION
          ----------     ---------------     --------------------------------
                            NO  IDENTIFIER  TABLE  ENTRIES


+[       START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
| K200I   K200D                                                           K200C
|%M00439 %M00440                                                        %Q00020
+--] [-----]/[-------------------------------------------------------------( )--
|
| << RUNG 4 >>
|
| K200X                                                                   K200I
|%M00434                                                                %M00439
+--] [---------------------------------------------------------------------( )--
|
+[  START  OF  LD  BLOCK FQ_106     ]
|
|       VARIABLE DECLARATIONS        ]
```

V A R I A B L E   D E C L A R A T I O N   T A B L E

REFERENCE	NICKNAME	REFERENCE DESCRIPTION
%L00001	THD	FLOW THRESHHOLD

I D E N T I F I E R   T A B L E

IDENTIFIER	IDENTIFIER TYPE	IDENTIFIER DESCRIPTION
	NO IDENTIFIER TABLE ENTRIES	

```
+[       START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|         +------+                                          +------+
+---------+ INT_ +------------------------------------------+ ADD_ +-
|         | TO_  |                                          | DINT |
|         | DINT |                                          |      |
|FII106   |      | FI106_D                          FI106_D |      | FQ106_D
|   )001--+IN   Q+-%R00031                         %R00031--+I1   Q+-%R00025
|         +------+                                          |      |
|                                                           |      |
|                                                  FQ106_D |      |
|                                                  %R00025--+I2    |
|                                                           +------+
|
+[        END OF BLOCK LOGIC        ]
|
+[  START  OF  LD  BLOCK FI_107    ]
|
|      VARIABLE DECLARATIONS       ]

          V A R I A B L E   D E C L A R A T I O N   T A B L E

     REFERENCE      NICKNAME          REFERENCE DESCRIPTION
     ---------      --------     --------------------------------
                      NO  VARIABLE  TABLE  ENTRIES


                   I D E N T I F I E R   T A B L E

     IDENTIFIER     IDENTIFIER TYPE         IDENTIFIER DESCRIPTION
     ----------     ---------------     --------------------------------
                      NO  IDENTIFIER  TABLE  ENTRIES


+[      START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|        +------+
+--------+ LT_  +-
|        | INT  |
|        |      |                                                            FAL107
|(  .07  |      |                                                            %M00029
|%AI0003-+I1   Q+-----------------------------------------------------------( )--
|        |      |
|F107_L  |      |
|%R00130-+I2    |
|        +------+
|
+[      END OF BLOCK LOGIC         ]
|
+[  START  OF  LD  BLOCK FQ_107    ]
|
|         VARIABLE DECLARATIONS        ]

              V A R I A B L E   D E C L A R A T I O N   T A B L E

           REFERENCE      NICKNAME            REFERENCE DESCRIPTION
           ---------      --------       --------------------------------
           %L00001        THD            FLOW THRESHHOLD

                          I D E N T I F I E R   T A B L E

           IDENTIFIER     IDENTIFIER TYPE         IDENTIFIER DESCRIPTION
           ----------     ---------------    --------------------------------
                          NO  IDENTIFIER  TABLE  ENTRIES

+[        START OF BLOCK LOGIC        ]
|
| << RUNG 3 >>
|
|          +-----+                                   +-----+
+---------+ INT_+-----------------------------------+ ADD_+-
|          | TO_ |                                   | DINT|
|          | DINT|                                   |     |
|FIT107    |     | FI107_D                   FI107_D |     | FQ107_D
|    )003--+IN  Q+-%R00033                  %R00033--+I1  Q+-%R00027
|          +-----+                                   |     |
|                                                    |     |
|                                           FQ107_D  |     |
|                                           %R00027--+I2   |
|                                                    +-----+
|
+[        END OF BLOCK LOGIC        ]
|
+[  START  OF  LD  BLOCK FIC_110  ]
|
|      VARIABLE DECLARATIONS       ]

           V A R I A B L E   D E C L A R A T I O N   T A B L E

        REFERENCE      NICKNAME          REFERENCE DESCRIPTION
        ---------      --------     --------------------------------
                           NO  VARIABLE  TABLE  ENTRIES


                          I D E N T I F I E R   T A B L E

        IDENTIFIER     IDENTIFIER TYPE          IDENTIFIER DESCRIPTION
        ----------     ---------------     --------------------------------
                          NO  IDENTIFIER  TABLE  ENTRIES


+[       START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|          +------+
+---------+ GT_  +-
|          | INT  |
|          |      |                                                     FAH110
|   110    |      |                                                     %M00090
|%AI0006--+I1   Q+-----------------------------------------------------( )--
|          |      |
|F110_H    |      |
|%R00114--+I2     |
|          +------+
|
| << RUNG 4 >>
|
|          +------+
+---------+ LT_  +-
|          | INT  |
|          |      |                                                     FAL110
|FIT110    |      |                                                     %M00089
|%AI0006--+I1   Q+-----------------------------------------------------( )--
|          |      |
|F110_L    |      |
|%R00103--+I2     |
|          +------+
|
| << RUNG 5 >>
|
|    OAS
|%.00101                  +------+
+--] [--+----------------+ PID_+-
|       |                | ISA  |
|P110BS |                |      |
|%I00102|       F110_S   |      | FCV110
+--] [--+       %R00202--+SP  CV+--%AQ0002
|                        |      |
|               FIT110   |      |
|               %AI0006--+PV    |
|                        |      |
|                      --+MAN   |
|                        |      |
|                      --+UP    |
|                        |      |
|                      --+DN    |
|                        |      |
|                        +------+
|                         FC110
|                        %R00440
|
+[        END OF BLOCK LOGIC        ]
|
+[  START  OF  LD  BLOCK FIC_220  ]
|
|      VARIABLE DECLARATIONS       ]

          V A R I A B L E   D E C L A R A T I O N   T A B L E

        REFERENCE      NICKNAME         REFERENCE DESCRIPTION
        ---------      --------     --------------------------------
                           NO  VARIABLE  TABLE  ENTRIES


                     I D E N T I F I E R   T A B L E

        IDENTIFIER     IDENTIFIER TYPE        IDENTIFIER DESCRIPTION
        ----------     ---------------     --------------------------------
                          NO  IDENTIFIER  TABLE  ENTRIES


+[       START OF BLOCK LOGIC        ]
|
| << RUNG 3 >>
|
|         +-----+
+---------+ GT_ +-
|         | INT |
|         |     |                                                      FAH220
|   220   |     |                                                      %M00038
|   0012--+I1  Q+-----------------------------------------------------( )--
|         |     |
|F220_H   |     |
|%R00123--+I2   |
|         +-----+
|
| << RUNG 4 >>
|
|         +-----+
+---------+ LT_ +-
|         | INT |
|         |     |                                                      FAL220
|FIT220   |     |                                                      %M00039
|%AI0012--+I1  Q+-----------------------------------------------------( )--
|         |     |
|F220_L   |     |
|%R00110--+I2   |
|         +-----+
|
| << RUNG 5 >>
|
|    )AS
|%.  0105                     +------+
+--] [--+------------------+ PID_+-
|       |                  | ISA |
|P220BS |                  |     |
|%I00106|       F220_S     |     | FCV220
+--] [--+       %R00207--+SP  CV+-%AQ0003
|                          |     |
|               FIT220     |     |
|               %AI0012--+PV    |
|                          |     |
|                        --+MAN  |
|                          |     |
|                        --+UP   |
|                          |     |
|                        --+DN   |
|                          |     |
|                          +------+
|                           FC220
|                          %R00520
|
+[       END OF BLOCK LOGIC        ]
|
+[  START  OF  LD  BLOCK FQ_240    ]
|
      VARIABLE DECLARATIONS        ]
```

V A R I A B L E   D E C L A R A T I O N   T A B L E

REFERENCE	NICKNAME	REFERENCE DESCRIPTION
%L00001	THD	FLOW THRESHHOLD

I D E N T I F I E R   T A B L E

IDENTIFIER	IDENTIFIER TYPE	IDENTIFIER DESCRIPTION
	NO IDENTIFIER TABLE ENTRIES	

```
+[       START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|         +-----+                                       +-----+
+---------+ INT_+---------------------------------------+ ADD_+-
|         | TO_ |                                       | DINT|
|         | DINT|                                       |     |
|FIT240   |     | FI240_D                      FI240_D  |     | FQ240_D
|    0009-+IN  Q+-%R00035                      %R00035-+I1   Q+-%R00029
|         +-----+                                       |     |
|                                                       |     |
|                                              FQ240_D  |     |
|                                              %R00029-+I2    |
|                                                       +-----+
|
+[       END OF BLOCK LOGIC       ]
|
+[  START  OF  LD  BLOCK LIC_110  ]
|
|       VARIABLE DECLARATIONS       ]

            V A R I A B L E   D E C L A R A T I O N   T A B L E

      REFERENCE      NICKNAME           REFERENCE DESCRIPTION
      ---------      --------      --------------------------------
                         NO  VARIABLE  TABLE  ENTRIES


                     I D E N T I F I E R   T A B L E

      IDENTIFIER     IDENTIFIER TYPE         IDENTIFIER DESCRIPTION
      ----------     ---------------     --------------------------------
                        NO  IDENTIFIER  TABLE  ENTRIES


+[      START OF BLOCK LOGIC      ]
|
| << RUNG 3 >>
|
|LSHH110                                                                   LAHH110
|%I00090                                                                   %M00072
+--]/[-----------------+----------------------------------------------------( )--
|                      |
|         +-----+      |
+-   -----+ GE_ +-     |
|         | INT |      |
|         |     |      |
|LIT110   |     |      |
|%AI0007--+I1  Q+------+
|         |     |
|L110_HH  |     |
|%R00128--+I2   |
|         +-----+
|
| << RUNG 4 >>
|
|LSH110                                                                    LAH110
|%I00091                                                                   %M00073
+--]/[-----------------+----------------------------------------------------( )--
|                      |
|         +-----+      |
+---------+ GT_ +-     |
|         | INT |      |
|         |     |      |
|LIT110   |     |      |
|%AI0007--+I1  Q+------+
|         |     |
|L110_H   |     |
|%R00113--+I2   |
|         +-----+
| << RUNG 5 >>
|
|   110                                                                         LAL110
|%I00094                                                                         %M00074
+--]/[----------------------+-----------------------------------------------------( )--
|                           |
|          +-----+          |
+----------+ LT_ +--        |
|          | INT |          |
|          |     |          |
|LIT110    |     |          |
|%AI0007--+I1   Q+----------+
|          |     |
|L110_L    |     |
|%R00102--+I2    |
|          +-----+
|
| << RUNG 6 >>
|
|P110AS
|%I00101                      +-----+
+--] [--+---------------------+ PID_+--
|       |                     | ISA |
|P110BS |                     |     |
|%I00102|           L110_S    |     | F110_S
+--] [--+          %R00200--+SP  CV+--%R00202
|                              |     |
|                   LIT110    |     |
|                  %AI0007--+PV     |
|                              |     |
|                            --+MAN  |
|                              |     |
|                            --+UP   |
|                              |     |
|                            --+DN   |
|                              |     |
|                              +-----+
|                               LC110
|                              %R00400

+[         END OF BLOCK LOGIC          ]
+[  START  OF  LD  BLOCK LIC_121   ]
|
|       VARIABLE DECLARATIONS       ]

           V A R I A B L E   D E C L A R A T I O N   T A B L E

      REFERENCE      NICKNAME          REFERENCE DESCRIPTION
      ---------      --------     --------------------------------
                      NO  VARIABLE  TABLE  ENTRIES


                     I D E N T I F I E R   T A B L E

      IDENTIFIER     IDENTIFIER TYPE        IDENTIFIER DESCRIPTION
      ----------     ---------------    ----------------------------------
                     NO  IDENTIFIER  TABLE  ENTRIES


+[        START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|         +------+
+---------+ GT_  +-
|         | INT  |
|         |      |                                                        LAH121
|    121  |      |                                                       %M00082
|%AI0015--+I1   Q+------------------------------------------------------( )--
|         |      |
|L121_H   |      |
|%R00118--+I2    |
|         +------+
|
| << RUNG 4 >>
|
|         +------+
+---------+ LT_  +-
|         | INT  |
|         |      |                                                        LAL121
|LIT121   |      |                                                       %M00083
|%AI0015--+I1   Q+------------------------------------------------------( )--
|         |      |
|L121_L   |      |
|%R00107--+I2    |
|         +------+
|
+[         END OF BLOCK LOGIC        ]
|
+[  START  OF  LD  BLOCK LIC_131   ]
|
|       VARIABLE DECLARATIONS      ]

            V A R I A B L E   D E C L A R A T I O N   T A B L E

         REFERENCE     NICKNAME        REFERENCE DESCRIPTION
         ---------     --------    --------------------------------
                        NO  VARIABLE  TABLE  ENTRIES


                        I D E N T I F I E R   T A B L E

         IDENTIFIER     IDENTIFIER TYPE       IDENTIFIER DESCRIPTION
         ----------     ---------------    --------------------------------
                       NO  IDENTIFIER  TABLE  ENTRIES


+[       START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|         +------+
+---------+ GT_  +-
|         | INT  |
|         |      |                                                          LAH131
|    131  |      |                                                          %M00084
|%R00017--+I1   Q+----------------------------------------------------------( )--
|         |      |
|L131_H   |      |
|%R00115--+I2    |
|         +------+
|
+[        END OF BLOCK LOGIC        ]
|
+[  START  OF  LD  BLOCK LIC_145  ]
¦
¦       VARIABLE DECLARATIONS       ]

              V A R I A B L E   D E C L A R A T I O N   T A B L E

          REFERENCE      NICKNAME          REFERENCE DESCRIPTION
          ---------      --------      --------------------------------
                             NO  VARIABLE  TABLE  ENTRIES


                           I D E N T I F I E R   T A B L E

          IDENTIFIER      IDENTIFIER TYPE        IDENTIFIER DESCRIPTION
          ----------      ---------------    --------------------------------
                            NO  IDENTIFIER  TABLE  ENTRIES


+[      START OF BLOCK LOGIC      ]
¦
¦ << RUNG 3 >>
¦
¦         +-----+
+---------+ GT_ +-
¦         ¦ INT ¦
¦         ¦     ¦                                                        LAH145
¦[illegible]145  ¦     ¦                                                        %M00085
¦%AI0019--+I1  Q+--------------------------------------------------------( )--
¦         ¦     ¦
¦L145_H   ¦     ¦
¦%R00116--+I2   ¦
¦         +-----+
¦
¦ << RUNG 4 >>
¦
¦         +-----+
+---------+ LT_ +-
¦         ¦ INT ¦
¦         ¦     ¦                                                        LAL145
¦LIT145   ¦     ¦                                                        %M00086
¦%AI0019--+I1  Q+---------+----------------------------------------------( )--
¦         ¦     ¦         ¦
¦         ¦     ¦         ¦                                              LCM145
¦L145_L   ¦     ¦         ¦                                              %M00027
¦%R00105--+I2   ¦         +----------------------------------------------(R)--
¦         +-----+
¦
```

```
| << RUNG 5 >>
|
|         +------+
+---------+ GT_  +-
|         | INT  |
|         |      |                                                     LCM145
|LIT145   |      |                                                     %M00027
|%AI0019--+I1   Q+----------------------------------------------------(S)--
|         |      |
|L145_M   |      |
|%R00131--+I2    |
|         +------+
|
+[        END OF BLOCK LOGIC          ]
|
+[  START  OF  LD  BLOCK LIC_220  ]
|
|       VARIABLE DECLARATIONS       ]

              V A R I A B L E   D E C L A R A T I O N   T A B L E

          REFERENCE     NICKNAME        REFERENCE DESCRIPTION
          ---------     --------    --------------------------------
                           NO  VARIABLE  TABLE  ENTRIES


                          I D E N T I F I E R   T A B L E

          IDENTIFIER    IDENTIFIER TYPE        IDENTIFIER DESCRIPTION
          ----------    ---------------    --------------------------------
                          NO  IDENTIFIER  TABLE  ENTRIES


+[      START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|        +------+
+--------+ GT_  +-
|        | INT  |
|        |      |                                                  LAHH220
|LIT220  |      |                                                  %M00093
|%AI0011-+I1   Q+----------------------------------------------------( )--
|        |      |
|L220_HH |      |
|%R00126-+I2    |
|        +------+
|
| << RUNG 4 >>
|
|        +------+
+--------+ GT_  +-
|        | INT  |
|        |      |                                                  LAH220
|LIT220  |      |                                                  %M00094
|%AI0011-+I1   Q+--------+-------------------------------------------( )--
|        |      |        |
|L220_H  |      |        |
|%R00124-+I2    |        |
|        +------+        |
|LSH220                  |
|%I00085                 |
+--]/[-------------------+
|
| << RUNG 5 >>
|
|
|        +-----+
+-  -----+ LT_ +-
|        | INT |
|        |     |                                                         LAL220
|LIT220  |     |                                                         %M00095
|%AI0011-+I1  Q+--------+--------------------------------------------------( )--
|        |     |        |
|L220_L  |     |        |
|%R00111-+I2   |        |
|        +-----+        |
|LSL220                 |
|%I00084                |
+--]/[------------------+
|
| << RUNG 6 >>
|
|P220AS
|%I00105                 +-----+
+--] [--+----------------+ PID_+-
|       |                | ISA |
|P220BS |                |     |
|%I00106|        L220_S  |     | F220_S
+--] [--+        %R00205-+SP  CV+-%R00207
|                        |     |
|                LIT220  |     |
|                %AI0011-+PV    |
|                        |     |
|                       -+MAN   |
|                        |     |
|                       -+UP    |
|                        |     |
|                       -+DN    |
|                        |     |
|                        +-----+
|                          LC220
|                         %R00480
|
+[      END OF BLOCK LOGIC       ]
|
+[  START  OF  LD  BLOCK LIC_240  ]
¦
¦       VARIABLE DECLARATIONS       ]

          V A R I A B L E   D E C L A R A T I O N   T A B L E

          REFERENCE     NICKNAME          REFERENCE DESCRIPTION
          ---------     --------     --------------------------------
                         NO  VARIABLE  TABLE  ENTRIES


                      I D E N T I F I E R   T A B L E

          IDENTIFIER     IDENTIFIER TYPE        IDENTIFIER DESCRIPTION
          ----------     ---------------    --------------------------------
                        NO  IDENTIFIER  TABLE  ENTRIES


+[       START OF BLOCK LOGIC       ]
¦
¦ << RUNG 3 >>
¦
¦        +-----+
+--------+ GT_ +-
¦        ¦ INT ¦
¦        ¦     ¦                                                            LAHH240
¦    240 ¦     ¦                                                            %M00098
¦%..0008-+I1  Q+----------------------------------------------------------( )--
¦        ¦     ¦
¦L240_HH ¦     ¦
¦%R00127-+I2   ¦
¦        +-----+
¦
¦ << RUNG 4 >>
¦
¦        +-----+
+--------+ GT_ +-
¦        ¦ INT ¦
¦        ¦     ¦                                                            LAH240
¦LIT240  ¦     ¦                                                            %M00099
¦%AI0008-+I1  Q+----------------------------------------------------------( )--
¦        ¦     ¦
¦L240_H  ¦     ¦
¦%R00125-+I2   ¦
¦        +-----+
¦
```

```
| << RUNG 5 >>
|
|          +------+
+----------+ LT_  +-
|          | INT  |
|          |      |                                                   LAL240
|LIT240    |      |                                                  %M00034
|%AI0008--+I1   Q+-----------------------------------------------------( )--
|          |      |
|L240_L    |      |
|%R00112--+I2    |
|          +------+
|
| << RUNG 6 >>
|
|          +------+
+----------+ LT_  +-
|          | INT  |
|          |      |                                                  LALL240
|LIT240    |      |                                                  %M00035
|%AI0008--+I1   Q+-----------------------------------------------------( )--
|          |      |
|L240_LL   |      |
|%R00100--+I2    |
|          +------+
|
+[        END OF BLOCK LOGIC        ]
|
+[  START  OF  LD  BLOCK LIC_211   ]
|
|      VARIABLE DECLARATIONS        ]

             V A R I A B L E   D E C L A R A T I O N   T A B L E

         REFERENCE     NICKNAME         REFERENCE DESCRIPTION
         ---------     --------     --------------------------------
                         NO  VARIABLE  TABLE  ENTRIES


                        I D E N T I F I E R   T A B L E

         IDENTIFIER    IDENTIFIER TYPE         IDENTIFIER DESCRIPTION
         ----------    ---------------     -------------------------------
                        NO  IDENTIFIER  TABLE  ENTRIES


+[       START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|         +------+
----------+ GT_  +-
|         | INT  |
|         |      |                                                 LAH211
|    11   |      |                                                 %M00091
|....0014-+I1   Q+-----------------------------------------------------( )--
|         |      |
|L211_H   |      |
|%R00120-+I2     |
|         +------+
|
| << RUNG 4 >>
|
|         +------+
----------+ LT_  +-
|         | INT  |
|         |      |                                                 LAL211
|LIT211   |      |                                                 %M00092
|%AI0014-+I1    Q+-----------------------------------------------------( )--
|         |      |
|L211_L   |      |
|%R00109-+I2     |
|         +------+
|
+[       END OF BLOCK LOGIC         ]
+[  START  OF  LD  BLOCK LIC_205   ]
|
|       VARIABLE DECLARATIONS       ]

              V A R I A B L E   D E C L A R A T I O N   T A B L E

         REFERENCE     NICKNAME          REFERENCE DESCRIPTION
         ---------     --------     --------------------------------
                          NO  VARIABLE  TABLE  ENTRIES


                         I D E N T I F I E R   T A B L E

         IDENTIFIER     IDENTIFIER TYPE        IDENTIFIER DESCRIPTION
         ----------     ---------------   --------------------------------
                         NO  IDENTIFIER  TABLE  ENTRIES

+[       START OF BLOCK LOGIC       ]
|
| << RUNG 3 >>
|
|LSH205                                                                    LAH205
|%I00113                                                                  %M00032
+--]/[----------------------------------------------------------------------( )--
|
|  RUNG 4 >>
|
|LSL205                                                                    LAL205
|%I00112                                                                  %M00031
+--]/[----------------------------------------------------------------------( )--
|
+[        END OF BLOCK LOGIC        ]
|
THE FOLLOWING BLOCKS ARE NOT PROGRAMMED:
          P_212
```

```
- no use     * explicit use     + implicit use     # explicit and implicit use

 ==============================================================================
==========  INPUT  (%I)  GLOBAL USAGE
===============================================================================
REF.     |64      |56      |48      |40      |32      |24      |16      |8      |1
ADDRESS  +------- +------- +------- +------- +------- +------- +------- +-------+
          There were no references used in the range: %I00001 - %I00064
%I00128 *---*--- --****-* *---*-** *****--* --**-**- **-**--- -------- --------
%I00192                                                             *- ---*****

===============================================================================
==========  OUTPUT  (%Q)  GLOBAL USAGE
===============================================================================
REF.     |64      |56      |48      |40      |32      |24      |16      |8      |1
ADDRESS  +------- +------- +------- +------- +------- +------- +------- +-------+
%Q00064                             ***----- -----*** -******* ----**** ********

===============================================================================
==========  INTERNAL  (%M)  GLOBAL USAGE
===============================================================================
REF.     |64      |56      |48      |40      |32      |24      |16      |8      |1
ADDRESS  +------- +------- +------- +------- +------- +------- +------- +-------+
%M00064 ---*-*-* -*-*-*-* -*-*-*** ******** **-***-- -------- -------- ------**
%M00128 -----*** -------- --****** ******** -******* --*****- **-*--** *-------
%M00192 -------- -------- -------- -------- -------- ***---** -------- ------*-
%M00256 -------- ------*- *------- -------- --***--- ---***-- --**---* *-------
' [illegible]20 *----*** ----*--- -------- -------- -------- -------- -------- --------
[illegible]384 **----** *----*** ----***- ---***-- --***--- -***---- ***----* **----**
%M00448 ***----* **----** *----*-- ----*--- ----**-- --*----- -*------ -**----*
%M00512                                   ** ----***- ---***-- --***--- -***----

===============================================================================
==========  REGISTER  (%R)  GLOBAL USAGE
===============================================================================
REF.     |50        |40        |30        |20        |10       |1
ADDRESS  +--------- +--------- +--------- +--------- +--------+
%R00050 ---------- -----*-*-* -*-*-*---- ---------- ----------
%R00100 *--------- ---------- ---------- ---------- ----------
%R00150 ---------- ---------* ********** *-*-****** **-*-*-***
%R00200 *--------- ---------- ---------- ---------- ----------
%R00250 ---------- ---------- ---------- ------**-* *--*-*--*-
%R00300 *--------- ---------- ---------- ---------- ----------
%R00350 ---------- ---------- ---------- ---------- --*--*--**
%R00400 *--------- ---------- ---------- ---------- ----------
%R00450 ---------- *--------- ---------- ---------- ----------
%R00500 ---------- ---------- *--------- ---------- ----------
%R00550 ---------- ---------- ---------- *--------- ----------
%R00600 *--------- ---------- ---------- ---------- *---------
```

```
===============================================================================
==========  ANALOG INPUT  (%AI)  GLOBAL USAGE
===============================================================================
RE. .   |50        |40        |30        |20        |10       |1
ADDRESS +--------- +--------- +--------- +--------- +--------+
%AI0050                                *- ****-***** **********

===============================================================================
==========  ANALOG OUTPUT  (%AQ)  GLOBAL USAGE
===============================================================================
REF.    |50        |40        |30        |20        |10       |1
ADDRESS +--------- +--------- +--------- +--------- +--------+
%AQ0050                                                  *-***

===============================================================================
==========  SYSTEM REFERENCE  (%S)  GLOBAL USAGE
===============================================================================
REF.    |64      |56      |48      |40      |32      |24      |16      |8      |1
ADDRESS +------- +------- +------- +------- +------- +------- +------- +------+
          There were no references used in the range: %S00001 - %S00064
%S00128        * -------- -------- -------- -------- -------- -------- --------
```

GE FANUC

GE Fanuc Automation

Programmable Control Products

Series 90™-70 Programmable Controller

Reference Manual

GE FANUC

GE Fanuc Automation

Programmable Control Products

Series 90™ -70 Programmable Controller

Reference Manual

GFK-0265G

May 1995

GFL-002

Warnings, Cautions, and Notes as Used in this Publication

Warning

Warning notices are used in this publication to emphasize that hazardous voltages, currents, temperatures, or other conditions that could cause personal injury exist in this equipment or may be associated with its use.

In situations where inattention could cause either personal injury or damage to equipment, a Warning notice is used.

Caution

Caution notices are used where equipment might be damaged if care is not taken.

Note

Notes merely call attention to information that is especially significant to understanding and operating the equipment.

Alarm Master	CIMSTAR	Helpmate	PROMACRO	Series Six
CIMPLICITY	GEnet	Logicmaster	Series One	Series 90
CIMPLICITY 90-ADS	Genius	Modelmaster	Series Three	VuMaster
CIMPLICITY PowerTRAC	Genius PowerTRAC	ProLoop	Series Five	Workmaster

Preface

This manual describes the system operation, fault handling, and Logicmaster 90-70 programming instructions for the Series 90™-70 programmable controller. The Series 90-70 PLC is a member of the Series 90™ family of programmable logic controllers from GE Fanuc Automation.

Revisions to This Manual

The Release 6 CPU and Logicmaster software offer major new capabilities and functionality which we have documented in this manual and the *Logicmaster™ 90-70 Programming Software User's Manual* (GFK-0263F). Changes made to this manual reflect the features of Release 6 (June 1995) Series 90-70 PLC. Additionally, corrections have been made where necessary. The following list describes the major revisions of this manual, as compared to the previous version (GFK-0265F):

- The Release 6 CPU has the new capability to run multiple programs (up to 16 user programs, one of which can be an RLD or SFC program, the rest being standalone C programs). Most of the technical information on this feature is presented in chapter 2 of this manual. To make the information concerning this new ability easier to understand, we completely reorganized chapter 2 and added many new tables and sections. For a complete understanding of all the options you now have available, we suggest that you read all of chapter 2.
- The previous versions included the ability to incorporate C blocks into an RLD program. This version is capable of handling multiple standalone C programs as well as blocks. For more information, refer to chapter 2.
- There is a new PLC Sweep Mode, called Microcycle Sweep Mode, which has certain advantages when dealing with multiple programs. Refer to page 2-49 and following in this manual for detailed technical information on this new mode, and refer to the *Logicmaster™ 90-70 Programming Software User's Manual (GFK-0263F)* for information on configuring the CPU to use Microcycle Sweep and for information on active sweep adjustment into Microcycle Sweep.
- There are four new Schedule Modes designed to give you control over multiple programs. For information on what these modes are and the advantages of each, refer to page 2-52 and following. Refer to chapter 2 of this manual for technical information on these modes, but refer to chapter 3, section 9, "Multiple Programs: Program Declarations," in the *Logicmaster™ 90-70 Programming Software User's Manual (GFK-0263F)* for instructions on how to assign and control them within Logicmaster.
- Interrupt handling is different when running multiple programs. Refer to page 2-65 and following for information on interrupt handling with user programs.
- Throughout this manual we have added text where appropriate denoting special concerns when using Microcycle Sweep Mode, when running multiple programs, or when using certain Schedule Modes.

- There is a new FIP device for the Series 90-70 PLC. Refer to page 2-82 for information about PLC I/O and the FIP device. Also, refer to appendix A, page A-16, for information about the sweep impact of FIP I/O and FIP Bus Controllers (FBC) on the Series 90-70 PLC.
- There are several new faults, most of which are related to accommodating multiple programs or the new schedule modes. Refer to chapter 3 for information on these new faults.
- As part of the reorganization of material in this manual, all fault information, including fault references, are now in chapter 3.
- There are several service requests which have new capabilities, as well as a new service request (#32), as listed below.
  - SVCREQ #15 (which has a new extended format), page 4-213 and following
  - SVCREQ #17, page 4-218 and following
  - SVCREQ #20 (which also has a new extended format), page 4-222 and following
  - SVCREQ #22, page 4-228 and following
  - SVCREQ #25, now called, "Disable/Enable EXE Block and Standalone C Program Checksums," page 4-231 and following
  - SVCREQ #26, page 4-232 and following
  - SVCREQ #32 (new), page 4-235 and following
- Numerous entries have been added to the index.

Content of This Manual

Revision G of this manual contains the following chapters and appendixes.

Chapter 1. Introduction: provides an overview of the Series 90-70 PLC system and the Series 90-70 instruction set.

Chapter 2. System Operation: describes certain system operations of the Series 90-70 PLC system. This includes a discussion of the PLC system sweep sequences, the system power-up and power-down sequences, clocks and timers, security, I/O, and fault handling. It also includes general information for a basic understanding of programming ladder logic.

Chapter 3. Fault Explanations and Correction: provides troubleshooting information for a Series 90-70 PLC system. It explains fault descriptions in the PLC fault table and fault categories in the I/O fault table.

Chapter 4. Series 90-70 Instruction Set: describes programming instructions available for the Series 90-70 PLC. The information in this chapter is arranged as sections that correspond to the main program function groups.

Appendix A. CPU Performance Data: lists the memory size in bytes and the execution time in microseconds for each programming instruction. Memory size is the number of bytes required by the function in a ladder diagram application program. Appendix A also contains timing information for other PC tasks which, when used in conjunction with the instruction timings, can be used to predict CPU sweep times.

Appendix B. Interpreting Fault Tables: describes how to interpret the message structure format when reading the fault tables using Logicmaster 90-70 software.

Appendix C. Instruction Mnemonics: lists mnemonics that can be typed to display programming instructions while searching or editing a program.

Appendix D. Memory Allocation: provides a worksheet for determining the total number of bytes of user data used and how much is still available for the user program.

Appendix E. Key Functions: lists the special keyboard assignments used for the Logicmaster 90 software.

Appendix F. Using Floating-Point Numbers: describes special considerations for using floating-point math operations.

Related Publications

Logicmaster™ 90-70 Programming Software User's Manual (GFK-0263).

Logicmaster™ 90-70 Important Product Information (GFK-0350).

Series 90™-70 Programmable Controller Installation Manual (GFK-0262).

Series 90™ Programmable Coprocessor Module and Support Software User's Manual (GFK-0255).

Series 90™ PCM Development Software (PCOP) User's Manual (GFK-0487).

C Programmer's Toolkit for Series 90™-70 PLCs User's Manual (GFK-0646).

Series 90™ Sequential Function Chart Programming Language User's Manual (GFK-0854).

MegaBasic™ Programming Language Reference Manual (GFK-0256).

CIMPLICITY™ 90-ADS Alphanumeric Display System User's Manual (GFK-0499).

CIMPLICITY™ 90-ADS Alphanumeric Display System Reference Manual (GFK-0641).

Alphanumeric Display Coprocessor Module Data Sheet (GFK-0521).

Series 90™-70 Genius I/O System User's Manual (GEK-90486-1).

Series 90™-70 Genius I/O Analog and Discrete Blocks User's Manual (GEK-90486-2).

Workmaster® II PLC Programming Unit Guide to Operation (GFK-0401).

Series 90™-70 Genius Bus Controller User's Manual (GFK-0398).

Series 90-70 FIP Bus Controller User's Manual (GFK-1038).

Guidelines for the Selection of Third-Party VME Modules (GFK-0448).

Series 90™ Ethernet Communications User's Manual (GFK-0868).

Series 90™ MAP 3.0 Communications User's Manual (GFK-0869).

We Welcome Your Comments and Suggestions

At GE Fanuc Automation, we strive to produce quality technical documentation. After you have used this manual, please take a few moments to complete and return the Reader's Comment Card located on the next page.

David Bruton
Sr. Technical Writer

READER'S COMMENTS

GFZ-0050A

We invite your comments and welcome suggestions to make this manual more useful.

Publication No. ____________ Date of Publication ____________ Today's Date ____________

GENERAL COMMENTS:

	Improve	Acceptable	Good	Excellent
Contents	☐	☐	☐	☐
Organization	☐	☐	☐	☐
Accuracy	☐	☐	☐	☐
Clarity	☐	☐	☐	☐
Completeness	☐	☐	☐	☐
Examples/Illustrations	☐	☐	☐	☐
Referencing/Indexing	☐	☐	☐	☐
Readability	☐	☐	☐	☐

DETAILED COMMENTS: (Correct, expand, etc. – Please be specific.)

Page No.	Comment

Other suggestions for improving this document: ____________

As compared to other manufacturers of a similar product, how would you rate this document ?

Superior ☐ Comparable ☐ Inferior ☐ Don't Know ☐

Commments: ____________

Are you interested in subscribing to a documentation update plan? Yes ☐ No ☐

APPLICATION INFORMATION:

Indicate the type of user/reader function that you most nearly represent:

☐ System Designer	☐ Programmer
☐ Distributor	☐ Maintenance
☐ OEM	☐ Operator
☐ Installation	☐ Other (Please Specify) ____________

Type of Equipment: ☐ Series 90-70 ☐ Series 90-30 ☐ Series 90-20 ☐ Series Six ☐ Series Five ☐ Series One ☐ Genius I/O ☐ Other ____________

Comments concerning your specific application: ____________

CUT ON DOTTED LINE

From:

Name:	
Title:	
Company:	
Address:	
City/State/Zip:	
Telephone:	

Fold Here

No Postage
Necessary
If Mailed
In The
United States

BUSINESS REPLY MAIL
FIRST CLASS MAIL PERMIT NO. 995 CHARLOTTESVILLE, VA

POSTAGE WILL BE PAID BY ADDRESSEE:

ATTENTION MANAGER TECHNICAL PUBLICATIONS
GE Fanuc Automation North America Inc
P O BOX 8106
CHARLOTTESVILLE VA 22907-6063

Fold Here

Chapter 1 Introduction

The Series 90™ -70 PLC is a member of the GE Fanuc Series 90™ PLC family of programmable logic controllers (PLCs). It is easy to install and configure, offers advanced programming features, and is designed for compatibility with other PLCs offered in the Series 90 family of PLCs. Through the use of the latest design and manufacturing technology, open architecture VME bus, and the ability to connect to Genius I/O and several CPU models, the Series 90-70 PLC provides a powerful, cost-effective platform for small applications through the very largest.

Software Architecture

The software architecture provides a platform upon which to build structured control programs. Programs may be built from many program blocks, each of which is related to a control function. Structured programs permit parallel development of a complete program as a collection of program blocks developed independently by many different individuals or OEMs. Structured programs are easier to understand and debug. A control program may be built of many smaller program blocks, each of which can relate to a specific machine function. This approach makes it easier to isolate and associate control logic with machine functions.

The programming language and representation are based on IEC working draft 65A standard. Eventual adoption of this standard will make it easier to create programs which can be understood globally. It establishes the Series 90-70 PLC in the vanguard of the movement toward recognized international standards.

Beginning with Release 6 of Logicmaster and Release 6 PLC CPUs, it is possible to incorporate multiple programs into a folder. All of these programs can be written in C or one program can be an RLD or SFC program with the remaining programs written in C. In addition, the Release 6 PLCs have built-in debugging capabilities for C programs and external blocks. For more information on this feature, refer to the *C Programmer's Toolkit* manual (GFK-0646C).

Terminology Used in This Manual

The following terms are used with their defined meanings throughout this manual:

User program: any user-generated code, i.e., an RLD program, an SFC program, or a standalone C program

Block: any RLD block, Parameterized Subroutine Block, or external block—an external block being either a C block or an C function block (CFBK)

Fault Handling

Faults are handled by a software alarm processor function which time-stamps and logs I/O and system faults in two tables (the PLC fault table and the I/O fault table). These tables can be displayed on the Logicmaster 90-70 programmer screen or uploaded to a host computer or other coprocessor. Application programs can also gain access to the fault information.

Software Installation and Hardware Configuration

If the Logicmaster 90-70 programming software is not yet installed, refer to chapter 2, "Operation," of the *Logicmaster 90-70 Programming Software User's Manual*, GFK-0263, for instructions. The user's manual also explains how to create, transfer, edit, and print programs.

Configuration is the process of assigning logical addresses, as well as other characteristics, to the hardware modules in the system. It may be done either before or after programming, using the configuration software which is part of the Logicmaster 90-70 software; however, it is recommended that configuration be done first. For information on configuration, refer to chapter 11, "I/O Configuration," in the *Logicmaster 90-70 Programming Software User's Manual*, GFK-0263.

Series 90-70 RLD/SFC Instruction Set

Programming consists of creating an application program for a PLC. Chapter 4 of this manual describes the programming instruction set used to create ladder logic programs for the Series 90™-70 programmable controllers. For corresponding functionality in C applications, refer to the *C Programmer's Toolkit for Series 90™-70 PLCs User's Manual* (GFK-0646B). For more information about SFC functionality, refer to the *Series 90™ Sequential Function Chart Programming Language User's Manual* (GFK-0854).

Contacts, Coils, and Links: see section 1 of chapter 4

The most basic elements of a program are relay functions, which are described in chapter 4, section 1, "Relay Functions." These contacts and coils represent machine inputs and outputs, and can be used to control the flow of logic through the program. They enable or prevent the execution of other program functions in a rung and indicate the states of outputs. The Logicmaster 90-70 software provides many types of contacts and coils for maximum programming flexibility.

Timers and Counters: see section 2 of chapter 4

For information about using on-delay, off-delay, or stopwatch-type timer, as well as up and down counters, refer to chapter 4, section 2, "Timers and Counters."

Math: see section 3 of chapter 4

Math functions include addition, subtraction, multiplication, division, modulo division, square root, absolute value, trigonometric, logarithmic/exponential, and radian conversions. These functions are explained in chapter 4, section 3, "Math Functions."

Each math function operates on two signed, unsigned, double precision integer or real numbers of the same type. If the numbers you are working with are not the same type (for example, if one is a signed integer and the other is an unsigned integer), you must first program one of the conversion functions (described in section 8) to make the input types match.

Convert Data to Another Type: see section 8 of chapter 4

Many program functions (like the math functions) operate on numbers which must be of the same type. Use the conversion functions, described in chapter 4, section 8, "Conversion Functions," if you need to change a number to word, double word, BCD, or signed, unsigned, double precision integer or real format.

Compare Two Numbers: see section 4 of chapter 4

To compare two numbers (which must be the same type), to see whether one is greater than, equal to, or less than the other, use one of the relational functions described in chapter 4, section 4, "Relational Functions."

Manipulate Bit Strings: see section 5 of chapter 4

Chapter 4, section 5, "Bit Operation Functions," contains information about data move and boolean operations on data that is in the form of bit strings:

1. To perform boolean operations (AND, OR, XOR, NOT) on two bit strings of the same length.
2. To create an output string that is a copy of an input bit string, but with its bits inverted, shifted, or rotated. Also, use the data move functions to clear an area of memory and fill it with typed data.
3. To locate one bit in a string and determine its state, or set it to 1 or 0. If a larger area of memory should be cleared, use one of the data move functions.
4. To compare two bit strings to see if they are the same.

To perform similar functions of the same type, use the data move functions described in chapter 4, section 6, "Data Move Functions." To move data in and out of tables organized as FIFO or LIFO tables, use the data table functions described in chapter 4, section 7, "Data Table Functions."

Move Data in Tables: see section 7 of chapter 4

To move data which is organized as FIFO tables (for example, items on an assembly line) or LIFO tables (for example, a stack of pallets), use the data table functions described in chapter 4, section 7, "Data Table Functions."

Move Data: see section 6 of chapter 4

Refer to chapter 4, section 6, "Data Move Functions," to create program logic that will:

1. Copy data to another location. Data is copied as individual bits.
2. Move a block of constants into memory.
3. Clear an area of discrete or register reference memory.
4. Shift data from one memory location to another, capturing the data that has been shifted out.
5. Perform a bit sequence shift through an array of bits.
6. Swap two bytes within a word, or two words within a double word (over a range of one or more words/doublewords).

Array Move and Search: see section 7 of chapter 4

To search for array values and compare them to a specified value or copy a specified number of data elements, use the table functions described in chapter 4, section 7, "Table Functions."

Do I/O and Suspend I/O: see section 9 of chapter 4

To change the normal I/O update of rack-mounted and Genius I/O modules in the system for one CPU sweep, use the DO I/O and SUS I/O functions described in chapter 4, section 9, "Control Functions." These functions are not used for VME boards on the VME bus.

Communicate with VME Boards: see section 6 of chapter 4

If the PLC system includes VME boards on the VME bus, special program instructions must be used for all communications with those boards, including I/O updates. VME boards will only communicate with the Series 90 PLC during the logic portion of CPU sweep, in response to special programming instructions. See chapter 4, section 6, "Data Move Functions."

Communicate with Other Modules: see section 6 of chapter 4

If the program must communicate with an intelligent module in the system (for example, to send data to a PCM), use a COMMREQ function. See chapter 4, section 6, "Data Move Functions."

Special Services from the PLC: see section 9 of chapter 4

Use the SVCREQ function described in chapter 4, section 9, "Control Functions," to:

1. Change/read the constant sweep timer.
2. Read window values.
3. Change the programmer communications window state and values.
4. Change the system communications window state and values.
5. Reserved.
6. Change/read the checksum task state and number of words to checksum.
7. Change/read the time-of-day clock state and values.
8. Reset the watchdog timer.
9. Read the sweep time from the beginning of the sweep.
10. Read the program name this block is in.
11. Read the PLC ID.
12. Read the PLC RUN state.
13. Shut down the PLC.
14. Clear the fault tables.
15. Read the last-logged fault table entry.
16. Read the elapsed time clock.
17. Mask/unmask I/O interrupts.
18. Read the I/O override status.
19. Set Run Enable/Disable.
20. Read fault tables.
21. Log a user-defined PLC fault.
22. Mask/unmask timed interrupts.
23. Read master checksum.
24. Disable/Enable EXE Block and Standalone C Program Checksums

25–31. Reserved

32. Suspend/Resume I/O Interrupt

Rung Explanation: see section 9 of chapter 4

To add rung comment text to the program, use the COMMENT function described in chapter 4, section 9, "Control Functions."

Control Functions: see section 9 of chapter 4

Use the MCR function to execute part of the program with negative logic, or the JUMP function to skip part of the program entirely. See chapter 4, section 9, "Control Functions," for information.

Call a Program Block: see section 9 of chapter 4

Use the CALL instruction to cause the program to jump directly to a named program block. See chapter 4, section 9, "Control Functions," for information.

Additional Reference Information in This Manual

Appendix A lists the memory size in bytes and the execution time in microseconds for each programming instruction described in chapter 4. Appendix A also contains timing information for other PC tasks which, when used in conjunction with the instruction timings, can be used to predict CPU sweep times.

Appendix B describes how to interpret the message structure format when reading the PLC and I/O fault tables.

Refer to appendix C for a listing of the instruction mnemonics used with Logicmaster 90-70 software.

Use the worksheet in appendix D to determine the total number of bytes of user data used and how much is still available for the user program.

Refer to appendix E for a listing of the special keyboard assignments used with Logicmaster 90-70 software.

Refer to appendix F for IEEE format when dealing with floating-point math operations.

Chapter 2 System Operation

This chapter describes certain system operations of the Series 90-70 PLC system. The table displayed below summarizes the content of each section in this chapter.

Section	Title	Description	Page
1	Basic PLC Sweep Summary	Describes the major steps in a typical PLC sweep, including application program task execution, Programmer Communications Window, system communications window, and Background Window.	2-2
2	User Reference Data	Describes user reference data, system status/fault references, and data types.	2-10
3	Program Organization	Describes the structure and use of RLD blocks, PSB blocks, external blocks, and standalone C programs.	2-24
4	PLC Sweep Modes and Program Scheduling Modes	**NORMAL SWEEP, CONSTANT SWEEP, CONSTANT WINDOW, MICROCYCLE,** and **STOP** mode are explained in this section. Also describes triggered interrupt blocks/programs and timed interrupts.	2-45
5	Run/Stop Operations	Describes the four modes of operation supported by the 90-70 PLC: Run/Outputs Enabled, Run/Outputs Disabled, Stop/IO Scan, Stop/No IO Scan.	2-68
6	Power-Up and Power-Down Sequences	Describes the three parts of system power-up, including power-up self-test, PLC operation initialization, and system configuration, the power-down sequence, and retention of data memory.	2-70
7	Clocks and Timers	Describes the elapsed time clock, time-of-day clock, and watchdog timers.	2-73
8	System Security	Describes protection level request from the programmer, including password assignment and block lock, OEM protection and password, and the write protect keyswitch.	2-75
9	Series 90-70 PLC I/O System	Lists Model 70 I/O modules. This section also describes I/O data mapping and diagnostic data.	2-78

Section 1: Basic PLC Sweep Summary

The user program(s) in the Series 90-70 PLC execute in a repetitive fashion until stopped by a command from the programmer or a command from another device. In addition to executing the user program(s), the PLC obtains data from input devices, sends data to output devices, performs internal housekeeping, services the programmer, services other communications, and performs self-tests. The sequence of operations necessary to execute these components one time is called a sweep. Four different sweep modes are supported by the 90-70 CPU, and are described in detail in section 4 of this chapter. This section describes each one of the sweep phases in detail.

Basic PLC Sweep

There are seven major phases in a typical PLC sweep as shown in the following figure:

Figure 2-1. Phases of a Typical PLC Sweep

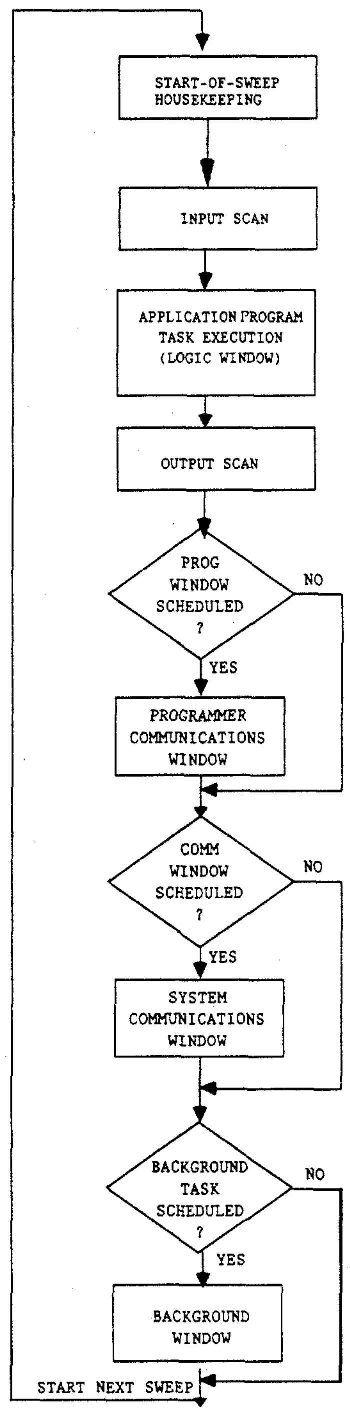

START-OF-SWEEP HOUSEKEEPING
INPUT SCAN
APPLICATION PROGRAM TASK EXECUTION (LOGIC WINDOW)
OUTPUT SCAN
PROG WINDOW SCHEDULED ?
NO
YES
PROGRAMMER COMMUNICATIONS WINDOW
COMM WINDOW SCHEDULED ?
NO
YES
SYSTEM COMMUNICATIONS WINDOW
BACKGROUND TASK SCHEDULED ?
NO
YES
BACKGROUND WINDOW
START NEXT SWEEP

Table 2-1. Major Phases in a Typical PLC Sweep

Step	Description
Housekeeping	Updating %S bits, determining timer update values, and determining the sweep mode occur in this phase.
Input Scan	The CPU reads input data from Bus controllers and input modules during this phase.
Application Program Task Execution (Logic Window)	The CPU solves the logic program(s), using data obtained from the input devices and setting bits to affect the state of output devices.
Output Scan	The CPU writes output data to Bus controllers and output modules during this phase. The user program checksum is computed during this phase of the sweep. Polling for faulted boards also occurs during this phase.
Programmer Communications Window	Communications with the programmer occur here with data and/or status transfer in both directions. In addition, reconfiguration of a module or a rack also occurs during this portion of the sweep.
System Communications Window	Communications with all intelligent devices—except the programmer—occur during this window. For example, supplying data to a PCM that is driving a process display would occur during this window.
Background Task Window	CPU self-tests occur in this window.

Housekeeping

The housekeeping portion of the sweep performs all of the tasks necessary to prepare for the start of the sweep. This includes updating %S bits, determining timer update values, and determining the mode of the sweep (**STOP** or **RUN**).

Input Scan

Scanning of the inputs occurs just prior to the logic solution. During the input scan, the CPU reads input data from the Genius Bus Controllers, FIP Bus Controllers, and from the Series 90-70 input modules.

Series 90-70 I/O modules are scanned from lowest to highest I/O reference address. There is no guaranteed scanning order for Bus Controllers.

Note

The input scan will not be performed if a program has an active SUS I/O function on the previous sweep.

Application Program Task Execution (Logic Window)

The logic window is the phase of the sweep where user programs execute.* Immediately following the completion of the input scan, the PLC Executive determines which user program(s) are to be run. Programs are then resumed and/or invoked as necessary. Solving the logic provides a new set of outputs.

*Interrupt programs and blocks can execute during any phase of the sweep. Refer to section 4 for further details.

There are many ways in which program execution can be controlled to meet the system's timing requirements. The following is a partial list of the commonly used methods.

- MCR and JUMP functions can be used to skip portions of the logic.
- The SUSIO function can be used to stop both the input scan and output scan for one sweep. I/O can be updated, as necessary, during the logic execution through the use of DOIO instructions.
- The SVCREQ function can be used to suspend or change the time allotted to the window portions of the sweep.
- Program logic can be structured so that blocks and programs are called more or less frequently, depending on their importance and on timing constraints.

Many of these program control capabilities described above are provided by the control functions, described in chapter 4, section 9, "Control Functions." A list of execution times for instructions can be found in appendix A.

Note

In Microcycle Sweep mode (refer to page 2-49), the logic window can be skipped or preempted as necessary by the PLC Executive.

Output Scan

Outputs are scanned immediately following logic solution. During the output scan, the CPU sends output data to the Genius Bus Controllers, the FIP Bus Controllers, and to the Series 90-70 output modules.

Series 90-70 output modules are scanned from lowest to highest I/O reference address. Bus Controllers are scanned from rack 0 to rack 7 and lowest to highest slot number within each rack.

Note

The output scan will not be performed if a program has an active SUS I/O function on the current sweep.

Polling for faulted boards also occurs during the output scan phase of the sweep. Faulted board polling recognizes replacement boards for faulted ones and reconfigures them into the system. If a board that was previously in the system or configured by the user to be in the system is listed as faulted, it must be polled periodically to determine if it has been replaced by a new board. Once a previously faulted board is detected as no longer faulted, reconfiguration is run in the Programmer Communications Window until the board(s) are reconfigured into the system.

The background checksum calculation also occurs during the output phase of the sweep. During each output scan phase of the sweep, the configured amount of words of user program are included in the checksum calculation. This checksum helps to ensure the integrity of the user logic while the CPU is executing. If the CPU is configured to perform a background checksum calculation (8 words is the default), then this part of output phase is performed; otherwise, it is skipped.

Programmer Communications Window

This part of the sweep is dedicated to communicating with the programmer and performing faulted board reconfiguration. This is also when communication with the C debugger occurs. If there is a programmer attached, a debugging session is active, or if there is a board in the system that requires reconfiguration (as detected during the faulted board polling portion of the sweep), the CPU executes the Programmer Window. The Programmer Window will not execute if there is no programmer attached, no active debug session occurring, and no board to be configured in the system. During the Programmer Window, highest priority is given to board configuration. Boards are configured as needed, up to the total time allocated to the Programmer Window.

The Programmer Window is used for communication between the CPU and the two dedicated programmer ports: the built-in SNP connection, and the parallel programmer connection. The CPU will complete any previously unfinished request and then begin to process any pending requests in the queue. When the time allocated for the window expires, processing stops.

The Programmer Window time defaults to 10 milliseconds. This value can be configured using the Logicmaster 90-70 Configuration software and stored to the PLC, or it can be changed online using the Logicmaster 90-70 Programming software.

Time and execution of the Programmer Window can also be dynamically controlled from the user program using SVCREQ function #3, described in chapter 4, section 9, "Control Functions." The Programmer Communications Window time can be set to a value from 0 to 255 milliseconds.

Note

Even if the Programmer Window is skipped, the PLC can still respond to commands to change mode or state, or to redefine the Programmer Window if the programmer is attached through the parallel port on the Bus Transmitter Module (BTM).

System Communications Window

The systems communications window is the part of the sweep used for communication between the CPU and intelligent modules such as the PCM, Genius Bus Controller, FIP Bus Controller, and ISO/Ethernet modules.

At the start of the systems communication window, the CPU will complete any previously unfinished request before executing any pending requests in the queue. When the time allocated for the window expires, processing stops.

The systems communication window defaults to "Run to Completion" mode. This means that all currently pending requests on all intelligent option modules are processed every sweep. If the tasks for the window are completed and there is time remaining, that time is given to the next window. A different mode can be configured using the Logicmaster 90-70 Configuration software and stored to the PLC, or it can be changed online using the Logicmaster 90-70 Programming software.

Time and execution of the system communications window can also be dynamically controlled from the user program using SVCREQ function #4, described in chapter 4, section 9, "Control Functions." This allows communications functions to be skipped during certain time-critical sweeps. The system communications window time can be set to a value from 0 to 255 milliseconds.

Background Window

A CPU self-test is performed in this window. Included in this self-test is a verification of the checksum for the 90-70 CPU operating system software.

The Background Window time defaults to 0 milliseconds. A different value can be configured using the Logicmaster 90-70 Configuration software and stored to the PLC, or it can be changed online using the Logicmaster 90-70 Programming software.

Time and execution of the Background Window can also be dynamically controlled from the user program using SVCREQ function #5, described in chapter 4, section 9, "Control Functions." This allows background functions to be skipped during certain time-critical sweeps.

Window Modes

The previous sections have described the phases of a typical PLC sweep. The Programmer Window, System Communications Window, and Background Window phases of the PLC sweep can be run in various modes, based on the PLC Sweep mode. (PLC sweep modes are described in detail in section 4.) The following three window modes are available:

Run-to-Completion In Run-to-Completion mode, all requests made when the window has started are serviced. When all pending requests in the given window have completed, the PLC will transition to the next phase of the sweep.

Constant In Constant Window mode, the total amount of time that the Programmer Communications, System Communications, and Background Window runs is fixed. If the time expires while in the middle of servicing a request, these windows are closed, and will be resumed the next sweep. If no requests are pending in this window, the PLC will cycle through these windows the specified amount of time polling for further requests.

Limited In Limited mode, the maximum time that the window runs is fixed. If the time expires while in the middle of servicing a request, the window is closed, and will be resumed the next time that the given window is run. If no requests are pending in this window, the PLC will proceed to the next phase of the sweep.

Data Coherency in Communications Windows

When running in Constant or Limited mode, the Programmer and System Communications Windows may be terminated early in all PLC sweep modes. If an external device is transferring a block of data, the coherency of the data block may be disrupted if the communications window is terminated prior to completing the request. The request will complete during the next sweep; however, part of the data will have resulted from one sweep and the remainder will be from the following sweep. When the PLC is in Normal Sweep mode and the Communication Window is in RUN-to-COMPLETION mode, the data coherency problem described above does not exist.

CPU Sweep in STOP Mode

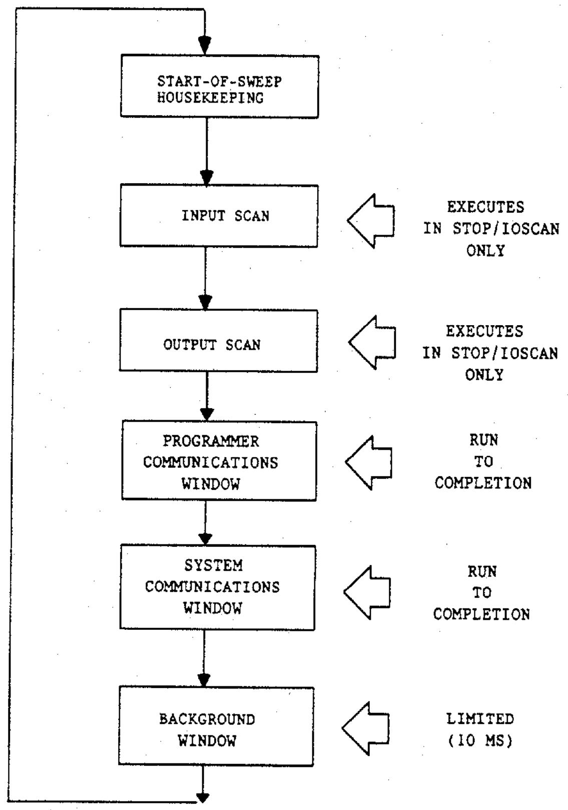
The 90-70 PLC has two modes of operation while it is in STOP mode, **STOP/NOIO** and **STOP/IOSCAN.**

When the PLC is in **STOP/NOIO** mode the Input Scan, Logic Window, and Output Scan phases of the PLC sweep are skipped.

When the PLC is in **STOP/IOSCAN** mode the Logic Window phase of the PLC is skipped but the Input Scan and Output Scan phases are performed each sweep.

In both **STOP/NOIO** and **STOP/IOSCAN** modes, the two Communications windows run in RUN-to-COMPLETION mode and the Background Window runs in LIMITED mode with a 10 millisecond limit.

Figure 2-2. CPU Sweep in STOP/NOIO and STOP/IOSCAN Mode


START-OF-SWEEP HOUSEKEEPING
INPUT SCAN
EXECUTES IN STOP/IOSCAN ONLY
OUTPUT SCAN
EXECUTES IN STOP/IOSCAN ONLY
PROGRAMMER COMMUNICATIONS WINDOW
RUN TO COMPLETION
SYSTEM COMMUNICATIONS WINDOW
RUN TO COMPLETION
BACKGROUND WINDOW
LIMITED (10 MS)

Note

STOP/IOSCAN is not supported in Microcycle Sweep mode.

PLC Sweep Modes

The 90-70 PLC supports four PLC sweep modes:

- Normal
- Constant Sweep
- Constant Window
- Microcycle

Normal Sweep

In Normal Sweep mode, each PLC sweep can consume a variable amount of time. The Logic Window is executed in its entirety each sweep. The Communications and Background Windows can be set to execute in a LIMITED or RUN-to-COMPLETION mode.

Constant Sweep

In Constant Sweep mode, each PLC sweep begins at a user specified Constant Sweep time after the previous PLC sweep began. The Logic Window is executed in its entirety each sweep. If there is sufficient time at the end of the sweep, the PLC will alternate among the Communications and Background Windows, allowing them to execute until it is time for the next sweep to begin.

Constant Window

In Constant Window mode, each PLC sweep can consume a variable amount of time. The Logic Window is executed in its entirety each sweep. The PLC will alternate among the Communications and Background Windows, allowing them to execute for a time equal to the user specified Constant Window timer.

Microcycle

In Microcycle Sweep mode, like Constant Sweep mode, each PLC sweep takes a fixed amount of time. The total sweep time (base cycle time) and the total time for the Communications and Background Windows is specified by the user. The Logic Window can be preempted in order to maintain the total sweep time and Communications and Background Window times. To satisfy the specified window times, the PLC alternates between the Programmer Communications Window, the System Communications Window, and the Background Window, allowing them to execute until it is time for the next sweep to begin.

Note

The information presented above summarizes the different sweep modes. For detailed information on PLC Sweep Modes, refer to the "PLC Sweep Modes and Program Scheduling Modes" section beginning on page 2-45.

Section 2: User Reference Data

User References

The PLC data used in an application program is stored as either discrete or register references.

Table 2-2. Register References

Type	Description
%R	Use the prefix %R to assign system register references which will store program data such as the results of calculations.
%AI	The prefix %AI represents an analog input register. This prefix is followed by the register address of the reference (e.g., %AI0015). An analog input register holds the value of one analog input or other value.
%AQ	The prefix %AQ represents an analog output register. This prefix is followed by the register address of the reference (e.g., %AQ0056). An analog output register holds the value of one analog output or other value.
%P*	Use the prefix %P to assign program register references which will store program data from the _MAIN block. This data can be accessed from all program blocks. The size of the %P data block is based on the highest %P reference in all blocks. (For more information, refer to the information on memory allocation in appendix D, "Memory Allocation.")
%L*	Use the prefix %L to assign local register references which will store program data unique to a block. The size of the %L data block is based on the highest %L reference in the associated block. (For more information, refer to the information on memory allocation in appendix D, "Memory Allocation.")

* These reference types are scoped at a program level, and therefore only visible to RLD programs.

Note

All register references are retained across a power cycle to the CPU.

Unbound References

Logicmaster programming software also supports the reference types %UR and %U to assign unbound references as place holders for actual references. Refer to section 12, "Editor Options," of chapter 3 of the *Logicmaster™ 90-70 Programming Software User's Manual*, GFK-0263, for further information.

Indirect References

You can use indirect referencing for all register references (%R, %AI, %AQ, %P, and %L) to identify a location in memory that contains the offset in the same memory type of the data to be used. Indirect references are entered in the same way as direct references, except that the @ character is used in place of the % character. For example, if %R00101 contains the value 1000, then @R00101 would instruct the PLC to use the data location of %R01000.

Indirect references can be useful when you want to perform the same operation to many registers. Use of indirect references can also be used to avoid repetitious ladder logic within the application program. It can be used in loop situations where each register is incremented by a constant or value specified until a maximum is reached.

Table 2-3. Discrete References

Type	Description
%I	The %I prefix represents input references. This prefix is followed by the reference's address in the input table (e.g., %I00121). %I references are located in the input status table, which stores the state of all inputs received from input modules during the last input scan. A reference address is assigned to discrete input modules using the configuration software. Until a reference address is assigned, no data will be received from the module. %I memory is always retentive.
%Q	The %Q prefix represents physical output references. The coil check function of Logicmaster 90-70 software checks for multiple uses of %Q references with relay coils or outputs on functions. Beginning with Release 4 of the software, you can select the level of coil checking desired (**SINGLE**, **WARN MULTIPLE**, or **MULTIPLE**). Refer to the "Editor Options" section of chapter 3 in the *Logicmaster 90-70 Programming Software User's Manual*, GFK-0263, for more information about this feature. The %Q prefix is followed by the reference's address in the output table (e.g., %Q00016). %Q references are located in the output status table, which stores the state of the output references as last set by the application program. This output status table's values are sent to output modules at the end of the program scan. A reference address is assigned to discrete output modules using the configuration software. Until a reference address is assigned, no data is sent to the module. A particular %Q reference may be either retentive or non-retentive. *
%M	The %M prefix represents internal references. The coil check function of Logicmaster 90-70 software checks for multiple uses of %M references with relay coils or outputs on functions. Beginning with Release 4 of the software, you can select the level of coil checking desired (**SINGLE**, **WARN MULTIPLE**, or **MULTIPLE**). Refer to the "Editor Options" section of chapter 3 in the *Logicmaster 90-70 Programming Software User's Manual*, GFK-0263, for more information about this feature. A particular %M reference may be either retentive or non-retentive. *
%T	The %T prefix represents temporary references. These references are never checked for multiple coil use and can, therefore, be used many times in the same program even when coil use checking is enabled. %T may be used to prevent coil use conflicts while using the cut/paste and file write/include functions. Because this memory is intended for temporary use, it is never retained through power loss or **RUN-TO-STOP-TO-RUN** transitions and cannot be used with retentive coils.
%S – %SC	The %S, %SA, %SB, and %SC prefixes represent system status references. These references are used to access special PLC data, such as timers, scan information, and fault information. For example, the %SC0012 bit can be used to check the status of the PLC fault table. Once the bit is set on by an error, it will not be reset until after the sweep. System status references include %S, %SA, %SB, and %SC references: • %S, %SA, %SB, and %SC can be used on any contacts. • %SA, %SB, and %SC can be used on retentive coils -(M)-. • %S can be used as word or bit-string input arguments to functions or function blocks. • %SA, %SB, and %SC can be used as word or bit-string input or output arguments to functions and function blocks. %S, %SA, %SB, and %SC references are non-retentive.
%G – %GE	The %G, %GA, %GB, %GC, %GD, and %GE prefixes represent global data references. These references are used to access data shared among several PLCs. %G – %GE references can be used on contacts and retentive coils because the memory is always retentive. %G – %GE cannot be used on non-retentive coils.

* Retentiveness is based on the type of coil. For more information, refer to "Retentiveness of Logic and Data" on page 2-14.

User Reference Size and Default

Maximum user references and default reference sizes for each model of CPU are listed in the tables below.

Table 2-4. User Reference Sizes

Item	CPU Model					
	925/915	924/914	788	780/781 782/789	771/772	731/732
Maximum %I reference	12288 points	12288 points	352 points	12288 points	2048 points	512 points
Maximum %Q reference	12288 points	12288 points	352 points	12288 points	2048 points	512 points
Maximum physical I/O (%I + %Q) *	12288 points*	12288 points*	352 points*	12288 points *	2048 points *	512 points*
Maximum %M reference	12288 points	12288 points	2048 points	12288 points	4096 points	2048 points
Maximum %T reference	256 points	256 points	256 points	256 points	256 points	256 points
%S total (S, SA, SB, SC)	512 points	512 points	512 points	512 points	512 points	512 points
%G (GA, GB, GC, GD, GE)	7680 points	7680 points	7680 points	7680 points	7680 points	1280 points
On-board User RAM	1024K bytes	512K bytes	None	None	None	32K bytes
Expansion Board RAM	None	None	512K bytes	512K bytes	512K bytes	None
Maximum %AI reference	8K words	8K words	8K words	8K words	8K words	8K words
Maximum %AQ reference	8K words	8K words	8K words	8K words	8K words	8K words
Maximum %R, 1K word increments	16K words	16K words	16K words	16K words	16K words	16K words
Maximum %L (per block)	8K words.	8K words	8K words	8K words	8K words	8K words
Maximum %P	8K words	8K words	8K words	8K words	8K words	8K words

* Total number of input points and output points together (size of %I + size %Q) cannot exceed this size.

Table 2-5. Default Memory Sizes

Memory Type	CPU Model					
	925/915	924/914	780/781/782 788/789	781/782	771/772	731/732
%AI	64 words	64 words	64 words	64 words	64 words	64 words
%AQ	64 words	64 words	64 words	64 words	64 words	64 words
%R	1024 words	1024 words	1024 words	1024 words	1024 words	1024 words
%P	0 words	0 words	0 words	0 words	0 words	0 words
%L	0 words	0 words	0 words	0 words	0 words	0 words

%G User References and CPU Memory Locations

The Series 90-70 CPU contains only one data space for all of the global data references (%G, %GA, %GB, %GC, %GD, and %GE). The internal CPU memory for this data is 7680 bits long. Logicmaster provides the user a subdivided representation by using %G, %GA, %GB, %GC, %GD, and %GE prefixes—allowing each of these prefixes to be used with bit offsets in the range 1−1280. Logicmaster interprets the requested global reference type (%G, %GA, %GB, %GC, %GD, or %GE) and converts it to the %G memory type and correct bit offset for use by the CPU. The actual mapping is shown in the table displayed below.

Table 2-6. %G References and Memory Locations

Global Data Type	%G	%GA	%GB	%GC	%GD	%GE
References Used by Logicmaster	%G1−1280	%GA1−1280	%GB1−1280	%GC1−1280	%GD1−1280	%GE1−1280
Memory Locations Used by the CPU	%G1−1280	%G1281−2560	%G2561−3840	%G3841−5120	%G5121−6400	%G6401−7680

This information is useful when programming 90-70 CPU application in the C language using the C Programmer's Toolkit. For more information about using the C Programmer's Toolkit, refer to the *C Programmer's Toolkit for Series 90™-70 PLCs User's Manual* (GFK-0646).

Note

A 731 CPU supports **only** %G since it has only 1280 points of global data.

Genius Global Data

The Series 90-70 PLC supports the sharing of data between multiple PLC systems that share a common Genius I/O bus. This mechanism provides a means for the automatic and repeated transfer of %G, %I, %Q, %AI, %AQ, and %R data. No special application programming is required to use global data since it is integrated into the I/O scan. All GE Fanuc PLCs that have Genius I/O capability can send global data to a Series 90-70 PLC and can receive global data from a Series 90-70 PLC. Logicmaster 90 configuration software is used to configure the receiving and transmitting of global data on a Genius I/O bus.

Transitions and Overrides

The %I, %Q, %M, and %G user references have associated transition and override bits. %T, %S, %SA, %SB, and %SC references have transition bits, but not override bits. The CPU uses transition bits for counters, transitional contacts, and transitional coils. Note that counters do not use the same kind of transition bits as contacts and coils. Transition bits for counters are stored within the locating reference.

Caution

Do not override transitional coils. If a transitional coil is overridden and the override is then removed, the coil will come on for one sweep. This can cause unexpected consequences in the PLC ladder logic and in field devices attached to the PLC.

When override bits are set, the associated references cannot be changed from the program or the input device; they can only be changed on command from the programmer.

Retentiveness of Logic and Data

Data is defined as retentive if it is saved by the PLC when the PLC is stopped. The Series 90-70 PLC preserves program logic, fault tables and diagnostics, checksums for program logic, overrides and output forces, word data (%R, %L, %P, %AI, %AQ), bit data (%I, %S, %G, fault locating references, and reserved bits), %Q and %M data (unless used with non-retentive coils), and word data stored in %Q and %M. %T data is not saved.

%Q and %M references are non-retentive (i.e., cleared when the PLC transitions from STOP to RUN, including power-up in RUN mode) whenever they are used with non-retentive coils. Non-retentive coils include coils -()-, negated coils -(/)-, SET coils -(S)-, and RESET coils -(R)-.

When %Q or %M references are used with retentive coils, or are used as function block outputs, the contents are retained through power loss and **RUN-TO-STOP-TO-RUN** transitions. Retentive coils include retentive coils -(M)-, negated retentive coils -(/M)-, retentive SET coils -(SM)-, and retentive RESET coils -(RM)-.

The last time a %Q or %M reference is programmed on a coil instruction determines whether the %Q or %M reference is retentive or non-retentive based on the coil type. For example, if %Q00001 was last programmed as the reference of a retentive coil, the %Q00001 data will be retentive. However, if %Q00001 was last programmed on a non-retentive coil, then the %Q00001 data will be non-retentive.

Note

When only standalone C programs are used, the retentive nature of data is based solely on the memory type since there are no coil instructions. In this case %Q and %M memory types are retentive.

Data Scope

Each of the user references has "scope"; that is, it may be available throughout the system, available to all programs, restricted to a single program, or restricted to local use within a block.

Table 2-7. Data Scope of User Reference Data

User Reference	Range	Scope
%I, %Q, %M, %T, %S, %SA, %SB, %SC, %G, %R, %AI, %AQ, convenience references, fault locating references	System	From any program, block, or host computer.
%P	Program	From any block, but not other programs.
%L	Local	Within a block.

In a RLD block:

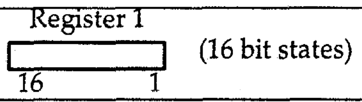
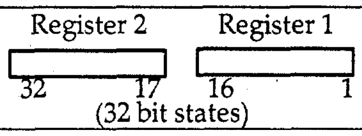
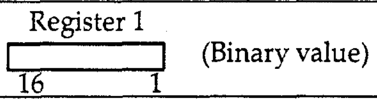
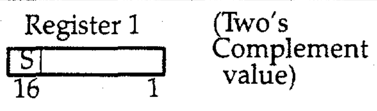
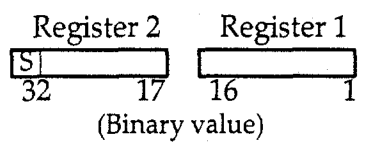
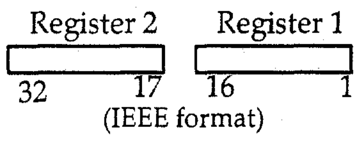
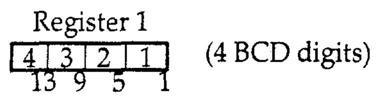
- %P should be used for program references that are shared with other blocks.
- %L are local references which can be used to restrict the use of register data to that block. These local references are not available to other parts of the program.

%I, %Q, %M, %T, %S, %SA, %SB, %SC, %G, %R, %AI, and %AQ references are available throughout the system.

Appendix D contains a Memory Allocation worksheet for determining the total number of bytes of user data used and how much is still available for the user program.

Data Types

Table 2-8. Data Types

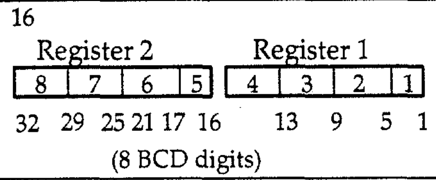
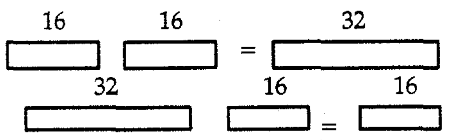
Type	Name	Description	Data Format
BIT	Bit	A Bit data type is the smallest unit of memory. It has two states, 1 or 0. A BIT string may have length N.	
BYTE	Byte	A Byte data type has an 8-bit value. It has 256 values (0–255). A BYTE string may have length N.	
WORD	Word	A Word data type uses 16 consecutive bits of data memory. The valid range of word values is 0000 hex to FFFF hex.	
DWORD	Double Word	A Double Word data type has the same characteristics as a single word data type, except that it uses 32 consecutive bits in data memory instead of only 16 bits.	
UINT	Unsigned Integer	Unsigned integers use 16-bit memory data locations. They have a valid range of 0 to +65535 (FFFF hex).	
INT	Signed Integer	Signed integers use 16-bit memory data locations, and are represented in 2's complement notation. The valid range of an INT data type is −32768 to +32767.	
DINT	Double Precision Integer	Double precision integers are stored in 32-bit data memory locations (actually two consecutive 16-bit memory locations) and are always signed values. (Bit 32 is the sign bit.) The valid range of a DINT data type is −2147483648 to +2147483867.	
REAL	Floating-Point	Real numbers use 32 consecutive bits (actually two consecutive 16-bit memory locations). The range of numbers that can be stored in this format is from ± 1.401298E-45 to ± 3.402823E+38. Refer to appendix F, "Using Floating-Point Math," for IEEE format.	
BCD-4	Four-Digit Binary Coded Decimal	Four-digit BCD numbers use 16-bit data memory locations. Each BCD digit uses four bits and can represent numbers between 0 and 9. This BCD coding of the 16 bits has a legal value range of 0 to 9999.	

S = Sign bit (0 = positive, 1 = negative).

Note

For a listing of ASCII characters, and their decimal and hexadecimal equivalents, refer to Tables 4-2 and 4-3 in chapter 4 in the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263.

Table 2-8. Data Types (continued)

Type	Name	Description	Data Format
BCD-8	Eight-Digit Binary Coded Decimal	Eight-digit BCD data types use two consecutive 16-bit data memory locations (32 consecutive bits). Each BCD digit uses 4 bits per digit to represent numbers from 0 to 9. The complete valid range of the 8-digit BCD data type is 0 to 99999999.	 16 Register 2 Register 1 8 7 6 5 4 3 2 1 32 29 25 21 17 16 13 9 5 1 (8 BCD digits)
MIXED	Mixed	A Mixed data type is available only with the MUL and DIV functions. The MUL function takes two integer inputs and produces a double integer result. The DIV function takes a double integer dividend and an integer divisor to product an integer results.	
ASCII	ASCII	8-bit encoded characters. A single reference is required to make up 2 (packed) ASCII characters. The first character of the pair corresponds to the low byte of the reference word. The remaining 7 bits in each section are converted as shown in Table 4-3 "ASCII Characters (Bit Pattern)" in chapter 4 of the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263. Command codes and non-displayable characters appear on the screen as non-alphanumeric characters (e.g., @).	

Note

Using function blocks that are not explicitly bit-typed will affect transitions for all bits in the written byte/word/dword. For further explanation, refer to the material about bit operations beginning on page 4-66.

Also for information about using floating-point numbers, refer to Appendix F, "Using Floating-Point Numbers."

System Status References

System status references (formerly called "Convenience" references) in the Series 90-70 PLC are assigned to %S, %SA, %SB, and %SC memory. Using them can save you quite a bit of time. They each have a system-supplied nickname which enables you to enter the nickname rather than exact %S reference. Examples of time tick references include T_10MS, T_100MS, T_SEC, and T_MIN. Examples of other convenience references include FST_SCN, ALW_ON, and ALW_OFF.

Note

%S bits are read-only bits; do not write to these bits.
You may, however, write to %SA, %SB, and %SC bits.

Listed below are available system status references which may be used in an application program. When entering logic, either the reference or the nickname can be used. Refer to chapter 3, "Fault Explanation and Correction," for more detailed fault descriptions and information on correcting the fault.

It is possible to use these special names in another context. However, if you attempt to use one of these names for some other use (e.g., block name, folder name, etc.), the Logicmaster 90-70 software displays this prompt:

```
Reuse system reserved nickname ________ ?  (Y/N)
```

Note

References not listed in the following table (e.g., %S0002) are not used for the Series 90-70 PLC.

Table 2-9. System Status References

Reference	Nickname	Definition
%S0001	FST_SCN	Current sweep is the first sweep in which the RLD or standalone C program executed. Set the first time the user program is executed after a **STOP/RUN** transition and cleared upon completion of its execution.
%S0003	T_10MS	0.01 second timer contact.
%S0004	T_100MS	0.1 second timer contact.
%S0005	T_SEC	1.0 second timer contact.
%S0006	T_MIN	1.0 minute timer contact.
%S0007	ALW_ON	Always ON.
%S0008	ALW_OFF	Always OFF.
%S0009	SY_FULL	Set when the PLC fault table fills up. Cleared when an entry is removed from the PLC fault table and when the PLC fault table is cleared.
%S0010	IO_FULL	Set when the I/O fault table fills up. Cleared when an entry is removed from the I/O fault table and when the I/O fault table is cleared.
%S0011	OVR_PRE	Set when an override exists in %I, %Q, %M, or %G memory.
%S0012	FRC_PRE	Force exists on a Genius point.
%S0013	PRG_CHK	Set when background program check is active.
%S0014	PLC_BAT	Set to indicate a bad battery in a Release 4 or later CPU. The contact is updated when a change in the battery status occurs.

Table 2-9. System Status References (cont'd)

Reference	Nickname	Definition
%S0121	FST_EXE	Current sweep is the first time this block has been called. Set when transitioning from **STOP** to **RUN**. FST_EXE is not available to standalone C programs.
%SA0001	PB_SUM	Set when a checksum calculated on the application program does not match the reference checksum. If the fault was due to a temporary failure, the discrete bit can be cleared by again storing the program to the CPU. If the fault was due to a hard RAM failure, then the CPU must be replaced.
%SA0002	OV_SWP	Set when the PLC detects that the previous sweep took longer than the time specified by the user. Cleared when the PLC detects that the previous sweep did not take longer than the specified time. It is also cleared during the transition from **STOP** to **RUN** mode. Only valid if the PLC is in **CONSTANT SWEEP** or **MICROCYCLE** mode.
%SA0003	APL_FLT	Set when an application fault occurs. Cleared when the PLC fault table is cleared.
%SA0009	CFG_MM	Set when a configuration mismatch is detected during system power-up or during a store of the configuration. Cleared by powering up the PLC when no mismatches are present or during a store of configuration that matches hardware.
%SA0010	HRD_CPU	Set when the diagnostics detects a problem with the CPU hardware. Cleared by replacing the CPU module.
%SA0011	LOW_BAT	Set when a low battery fault occurs. Cleared by replacing the battery and ensuring that the PLC powers up without the low battery condition.
%SA0012	LOS_RCK	Set when an expansion rack stops communicating with the PLC CPU. Cleared by fixing the problem and power cycling the rack.
%SA0013	LOS_IOC	Set when a Bus Controller stops communicating with the PLC. Cleared by replacing the module and cycling power on the rack containing the module.
%SA0014	LOS_IOM	Set when an I/O module stops communicating with the PLC CPU. Cleared by replacing the module and cycling power on the rack containing the module.
%SA0015	LOS_SIO	Set when an option module stops communicating with the PLC CPU. Cleared by replacing the module and cycling power on the rack containing the module.
%SA0017	ADD_RCK	Set when an expansion rack is added to the system. Cleared by cycling power on the rack containing the module and when the configuration matches the hardware after a store.
%SA0018	ADD_IOC	Set when a Bus Controller is added to a rack. Cleared by cycling power on the rack containing the module and when the configuration matches the hardware after a store.
%SA0019	ADD_IOM	Set when an I/O module is added to a rack. Cleared by cycling power on the rack containing the module and when the configuration matches the hardware after a store.

Table 2-9. System Status References (cont'd)

Reference	Nickname	Definition
%SA0020	ADD_SIO	Set when an option module is added to a rack. Cleared by cycling power on the rack containing the module and when the configuration matches the hardware after a store.
%SA0022	IOC_FLT	Set when a Bus Controller reports a bus fault, a global memory fault, or an IOC hardware fault. Cleared by cycling power on the rack containing the module and when the configuration matches the hardware after a store.
%SA0023	IOM_FLT	Set when an I/O module reports a circuit or module fault. Cleared by cycling power on the rack containing the module and when the configuration matches the hardware after a store.
%SA0027	HRD_SIO	Set when a hardware failure is detected in an option module. Cleared by replacing the module and cycling power on the rack containing the module.
%SA0029	SFT_IOC	Software failure in the I/O Controller.
%SA0031	SFT_SIO	Set when a Genius Bus Controller detects an internal software error. Cleared by cycling power on the main rack and when the configuration matches the hardware.
%SA0032	SBUS_ER	Set when a bus error occurs on the VME bus backplane. Cleared by cycling power on the main rack.
%SA0081 – %SA0112		Set when a user-defined fault is logged in the PLC fault table. For more information on user-defined fault logging, refer to the description of SVCREQ function #21 on page 4-226. These bits are cleared when the PLC fault table is cleared.
%SB0001	WIND_ER	Set when there is not enough time to start the Programmer Window in **CONSTANT SWEEP** or **MICROCYCLE** mode, or when there is not enough time to start the logic window in **MICROCYCLE** mode. Cleared when the PLC detects that the previous sweep did have enough time to perform the window.
%SB0009	NO_PROG	Set when the PLC CPU powers up with memory preserved, but no user program is present. Cleared when the PLC powers up with a program present.
%SB0010	BAD_RAM	Set when the CPU detects corrupted RAM memory at power-up. Cleared when the CPU detects that RAM memory is valid at power-up.
%SB0011	BAD_PWD	Set when a password access violation occurs. Cleared when the PLC fault table is cleared.
%SB0012	NUL_CFG	Set when an attempt is made to put the PLC in **RUN** mode when there is no configuration data present. Cleared when configuration data is present and the PLC is put in **RUN** mode.
%SB0013	SFT_CPU	Set when the CPU detects an error in the CPU operating system software. Cleared by cycling power to the CPU.
%SB0014	STOR_ER	Set when an error occurs during a programmer store operation. Cleared when a store operation is completed successfully.
%SB0016	MAX_IOC	Set when more than 32 IOCs are configured for the system. Cleared by modifying the configuration and storing to the PLC CPU.
%SB0017	SBUS_FL	Set when the PLC fails to gain access to the bus. Cleared by cycling power on the main rack.

Table 2-9. System Status References (cont'd)

Reference	Nickname	Definition
%SC0009	ANY_FLT	Set when any fault occurs. Cleared when both fault tables are cleared.
%SC0010	SY_FLT	Set when any fault occurs that causes an entry to be placed in the PLC fault table. Cleared when the PLC fault table is cleared.
%SC0011	IO_FLT	Set when any fault occurs that causes an entry to be placed in the I/O fault table. Cleared when the I/O fault table is cleared.
%SC0012	SY_PRES	Set as long as there is at least one entry in the PLC fault table. Cleared when the PLC fault table is cleared. Once the bit is set on by an error, it will not be reset until after the sweep.
%SC0013	IO_PRES	Set as long as there is at least one entry in the I/O fault table. Cleared when the I/O fault table is cleared.
%SC0014	HRD_FLT	Set when a hardware fault occurs. Cleared when both fault tables are cleared.
%SC0015	SFT_FLT	Set when a software fault occurs. Cleared when both fault tables have are cleared.

Other References

The fault references are discussed in chapter 3 of this manual but are presented here for your convenience.

Other References Available to the User

System Fault Reference	Description	Page Number
ANY_FLT	Any fault in the system.	3-2
SY_FLT	Any system fault in the Series 90 PLC.	
IO_FLT	Any I/O fault.	
SY_PRES	Indicates a new entry in the PLC fault table.	
IO_PRES	Indicates a new entry in the I/O fault table.	
HRD_FLT	Any hardware fault.	
SFT_FLT	Any software fault.	

Configurable Fault References (Default Action)	Description	Page Number
SBUS_ER (diagnostic)	System bus error. (The BSERR* signal was generated on the VME system bus.)	3-4
HRD_CPU (fatal)	PLC CPU hardware fault, such as failed memory device or failed serial port).	3-4
HRD_SIO (diagnostic)	Non-fatal hardware fault on any module in the system, such as failure of a serial port on a PCM.	3-4
SFT_IOC (diagnostic)	Non-recoverable software error in a Genius Bus Controller.	3-4
SFT_SIO (diagnostic)	Non-recoverable software error in a PCM or LAN interface module.	3-4

Configurable Fault References (Default Action)	Description	Page Number
PB_SUM (fatal)	Program or block checksum failure during power-up or in **RUN** mode.	3-4
LOW_BAT (diagnostic)	Low battery signal from CPU or another module in system.	3-4
OV_SWP (diagnostic)	Constant sweep time exceeded.	3-4
SY_FULL IO_FULL (diagnostic)	PLC fault table full (16 entries). I/O fault table full (32 entries).	3-4
APL_FLT (diagnostic)	Application fault.	3-5
LOS_RCK (diagnostic)	Loss of rack (BRM failure, loss of power), or missing a configured rack.	3-5
LOS_IOC (diagnostic)	Loss of Bus Controller channel, or missing a configured Bus Controller.	3-5
LOS_IOM (diagnostic)	Loss of I/O module (does not respond), or missing a configured I/O module.	3-5
LOS_SIO (diagnostic)	Loss of option module (does not respond), or missing a configured module.	3-5
ADD_RCK (diagnostic)	New rack added, or previously faulted rack has returned.	3-5
ADD_IOC (diagnostic)	Previously faulted Bus Controller is no longer faulted.	3-5
ADD_IOM (diagnostic)	Previously faulted I/O module is no longer faulted.	3-5
ADD_SIO (diagnostic)	New option module is added, or previously faulted module no longer faulted.	3-5
IOC_FLT (diagnostic)	Non-fatal bus or Bus Controller error, more than 10 bus errors in 10 seconds (error rate is configurable).	3-5
IOM_FLT (diagnostic)	Point or channel on an I/O module; a partial failure of the module.	3-5
CFG_MM (fatal)	Wrong module type detected during power-up or **RUN** mode. The PLC does not check the configuration parameters set up for individual modules such as Genius I/O blocks.	3-5

Non-Configurable Faults	Description	Result
SBUS_FL (fatal)	System bus failure. The PLC CPU was not able to access the VME bus. BUSGRT*NMI error.	3-6
NO_PROG (information)	No application program is present at power-up. Should only occur the first time the PLC is powered up or if the battery-backed RAM containing the program fails.	3-6
BAD_RAM (fatal)	Corrupted program memory at power-up. Program could not be read and/or did not pass checksum tests.	3-6
WIND_ER (information)	Window completion error. Servicing of Programmer or Logic Window was skipped. Occurs in **CONSTANT SWEEP** or **MICROCYCLE SWEEP** mode.	3-6
BAD_PWD (information)	Change of privilege level request to a protection level was denied; bad password.	3-6
NUL_CFG (fatal)	No configuration present upon transition to **RUN** mode. Running without a configuration is equivalent to suspending the I/O scans.	3-6
SFT_CPU (fatal)	CPU software fault. A non-recoverable error has been detected in the CPU. May be caused by Watchdog Timer expiring.	3-6
MAX_IOC (fatal)	The maximum number of bus controllers has been exceeded. The Series 90 PLC supports 32 bus controllers.	3-6
STOR_ER (fatal)	Download of data to PLC from the programmer failed; some data in PLC may be corrupted.	3-6

Note

Fault locating references are discussed on pages 3-7 through 3-8 in this manual.

FIP locating references are discussed on pages 3-9 through 3-10 in this manual.

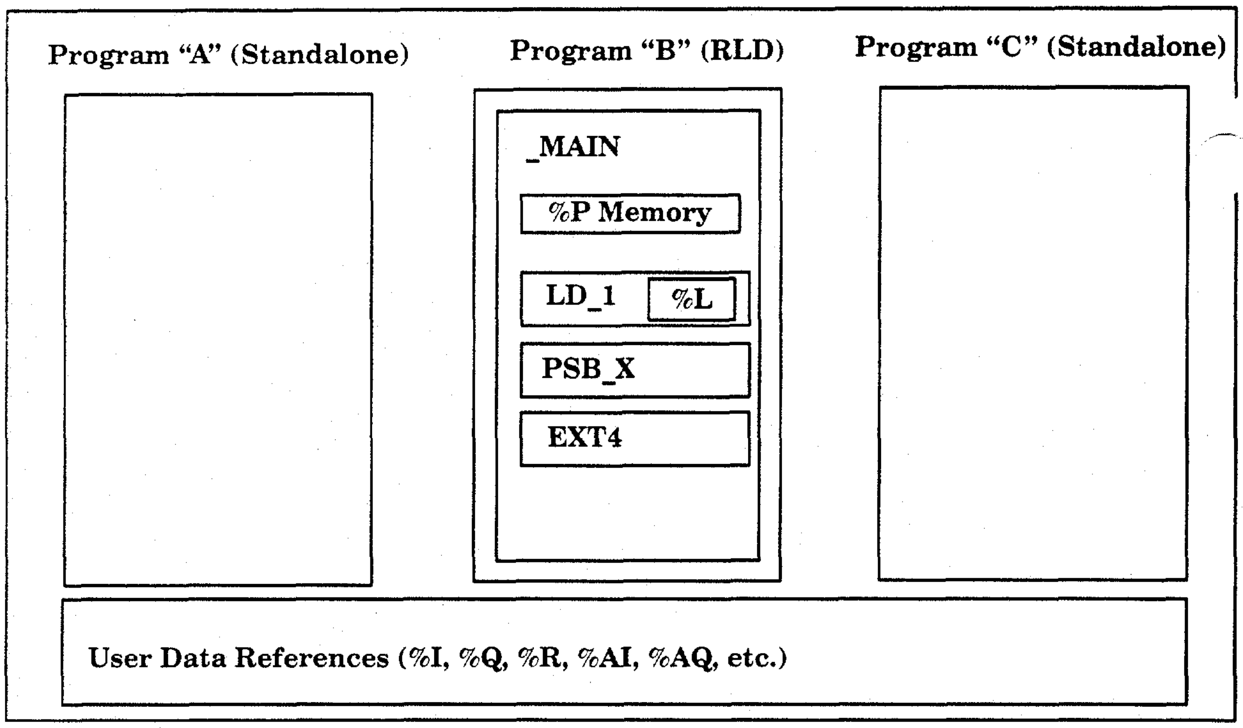
Refer to the *Series 90™ Sequential Function Chart Programming Language User's Manual* (GFK-0854) for SFC references.

Unbound (%U) References are not bound to the PLC but used within Logicmaster only; for that reason they are discussed in the Logicmaster User's Manual rather than in this manual. Refer to section 12, "Editor Options," in chapter 3 of the *Logicmaster™ 90-70 Programming Software User's Manual*, GFK-0263, for further information

Section 3: Program Organization

The user program(s) contains the logic that is used to process input data and control output data. Program logic is executed repeatedly by the PLC. The Series 90-70 PLC allows up to 16 user programs, with a maximum of 1 RLD program. Refer to tables 2-4 and 2-5 on page 2-12 for a listing of program sizes and reference limits for each CPU model.

The following figure depicts 3 user programs, two of which are standalone C programs. The RLD program consists of four blocks (_MAIN, LD_1, PSB_X, and EXT4). The figure further illustrates the scoping of various memory types: all references except %P and %L are visible to the standalone programs; %P memory is visible to all of the program blocks, and the LD_1 block has its own local data, %L. Details of standalone programs and blocks are described in the following sections.



Ladder Logic Programming

A RLD program for the Series 90-70 PLC consists of one or more units called blocks. Four types of blocks are supported by the Series 90-70 PLC:

	Programming Language	Size Limit	Number of Parameter Pairs	Notes
LD	Ladder Logic	16k bytes	n/a	
SFC	Ladder Logic/ SFC	16k bytes	n/a	SFC blocks cannot be used as interrupt blocks.
PSB	Ladder Logic	16k bytes	0–7	
External	C	64,000 by- tes	0–7	External blocks cannot call any other blocks. External blocks are created using the C pro- grammer's Toolkit.

Table 2-10. Block Types

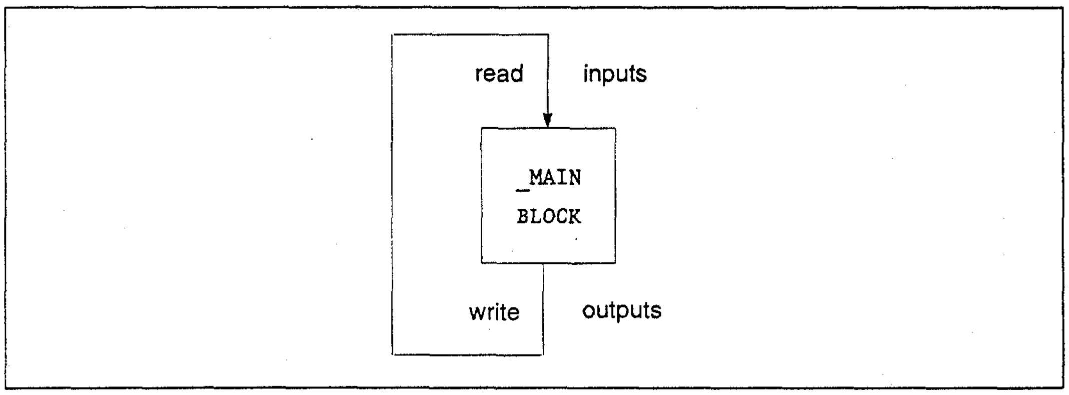
SFC programming is described in detail in the *Series 90 Sequential Function Chart Programming Language User's Manual,* GFK-0854.

Note

Up to 255 blocks can be used. The maximum number of block calls that can be programmed within a given block is 64. The maximum number of programmed calls to a particular block is 255. (A block can be executed any number of times, but there cannot be more than 255 explicit calls to any given block.)

Main Block

When using an RLD program there is always a _MAIN block. RLD program execution begins with the _MAIN block.

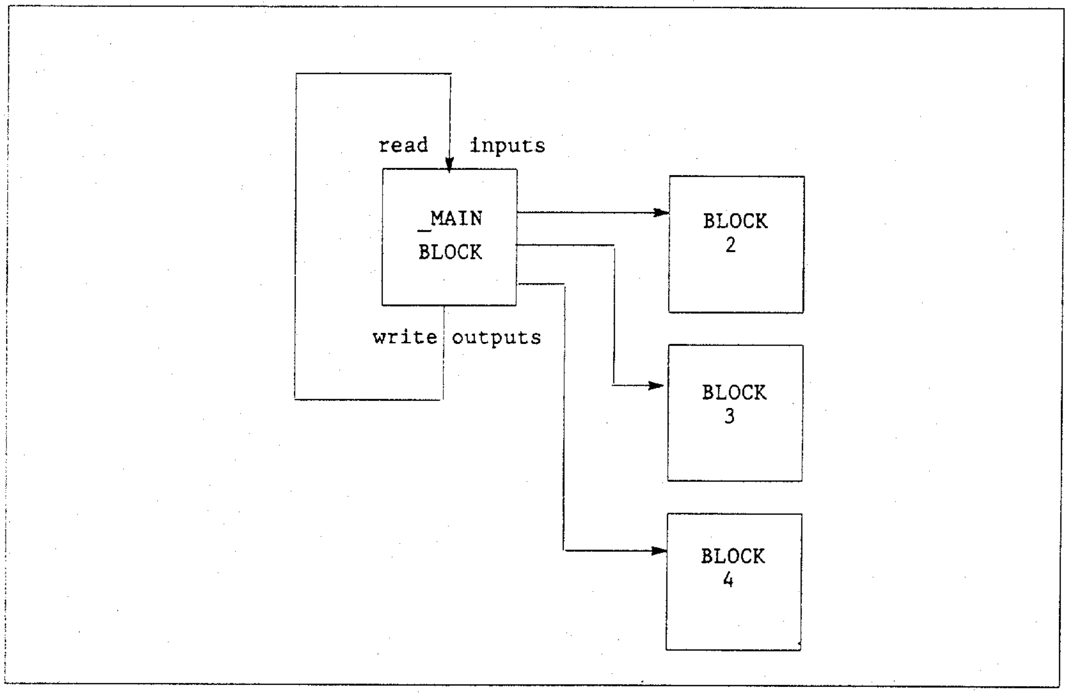


Blocks

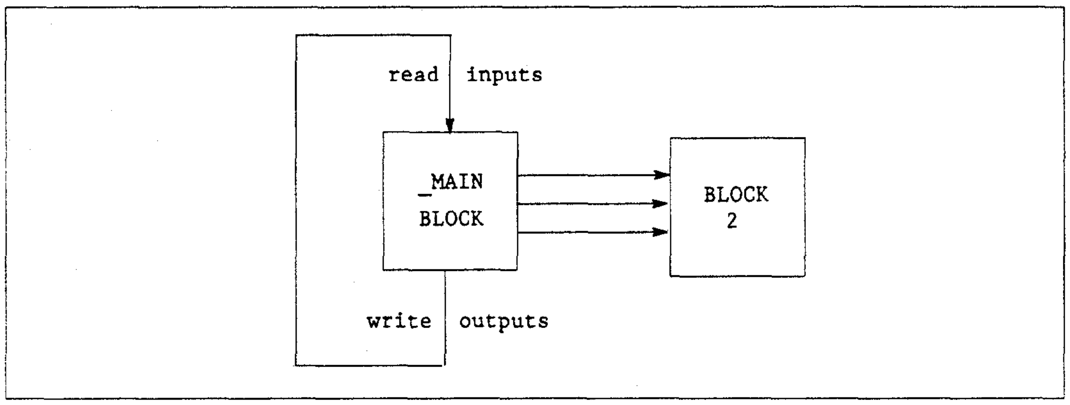
Structuring a program as blocks enables you to re-use logic. Logic which needs to be repeated can be entered in a block. Calls would then be made to that block to execute the logic. In this way, total program size is reduced. Dividing a program into smaller blocks also simplifies programming and reduces the overall amount of logic needed for the program.

Examples of Using Blocks

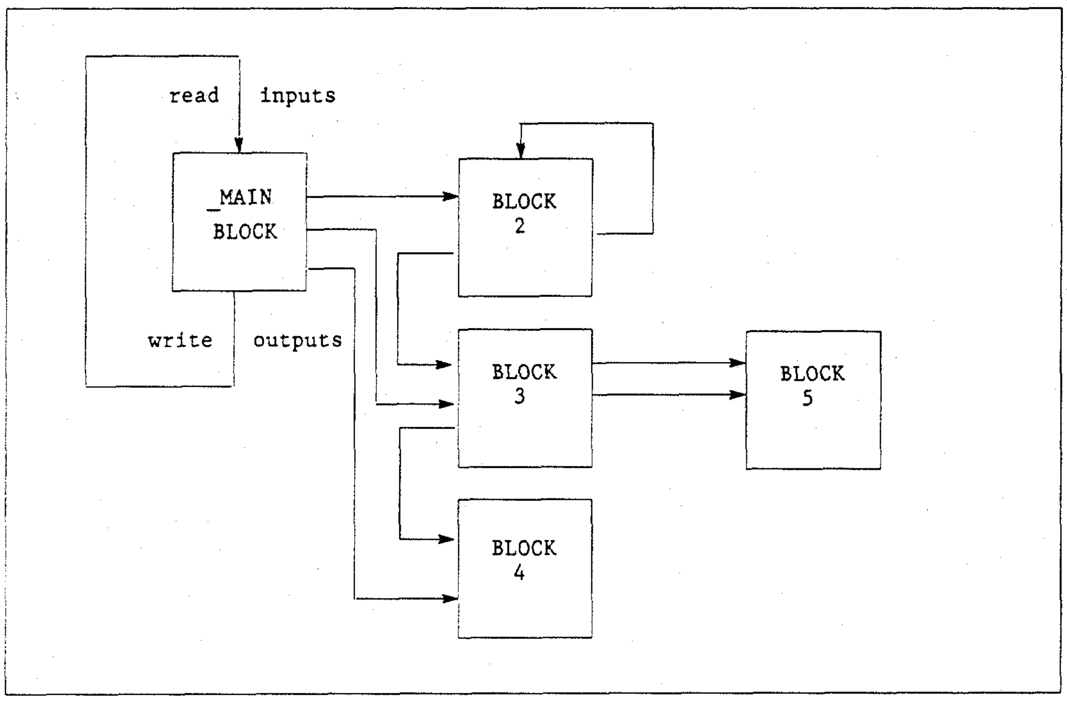
As an example, the logic for an RLD program could be divided into three blocks, each of which could be called as needed from the _MAIN block. (A block cannot call the _MAIN block.) In this example, the _MAIN block might contain little logic, serving primarily to sequence the other blocks.



A block can be used many times as the program executes. Logic which needs to be repeated several times in a program could be entered in a block. Calls would then be made to that block to access the logic.



In addition to being called from the _MAIN block, blocks can also be called by other blocks. A block may even call itself.



There is no limit to the number of levels of calls to blocks that Logicmaster 90 software will allow. However, the PLC will only allow a certain number of nested calls before an "Application Stack Overflow" fault is logged and the PLC transitions to **STOP/FAULT** mode. The call depth is guaranteed to be at least four on the 731 and 732 CPUs and eight on all other models. The actual call depth allowed depends on the amount of data (non-boolean) flow used in the blocks. If less than the 171 word data flow limit is used, then more nested calls may be made. The call level nesting counts the _MAIN block as level 1. The illustration above shows three levels of calls.

Note

Before a block can be used, you need to define it in the block declarations. For information on block declarations, refer to the "Program Blocks, External Blocks, and Interrupts" section of chapter 3, "Program Editing," of the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263.

How Blocks Are Called

A block executes when called from the program logic in the _MAIN block or another block.

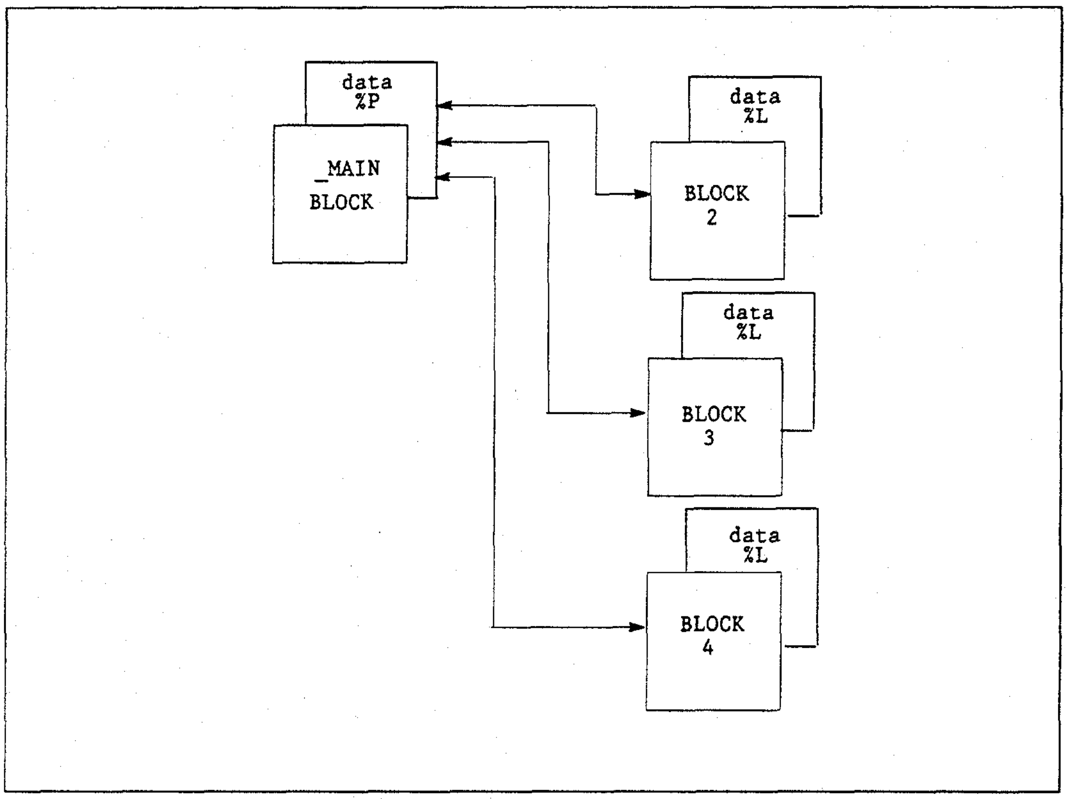
```
|   _____________
|--| CALL  PBK1  |
|  |_____________|
|
|%I00500 %Q00076  _______________
|--| |------| |---| CALL   PBK2  |
|                 |______________|
|
```

In the example above, PBK1 will always be called. Conditional logic can be used to control calling the block. In order for PBK2 to be called, both input %I00500 and output %Q00076 must be ON.

Blocks and Local Data

Each block in the RLD program can have an associated data block. The _MAIN data block is referenced by %P; all other data blocks are referenced by %L.

The size of the data block is dependent on the highest reference in its block for %L and in all blocks for %P. Appendix D, "Memory Allocation," provides a worksheet for determining the total number of bytes of user data used and how much is still available for the user program.



All blocks within the RLD program can use data associated with the _MAIN block (%P). Blocks can use their own %L references as well as the %P references that are available to all blocks. The _MAIN block cannot use %L.

Note

External blocks and Parameterized Subroutine Blocks do not have their own %L data; instead they inherit the %L data of the calling block.

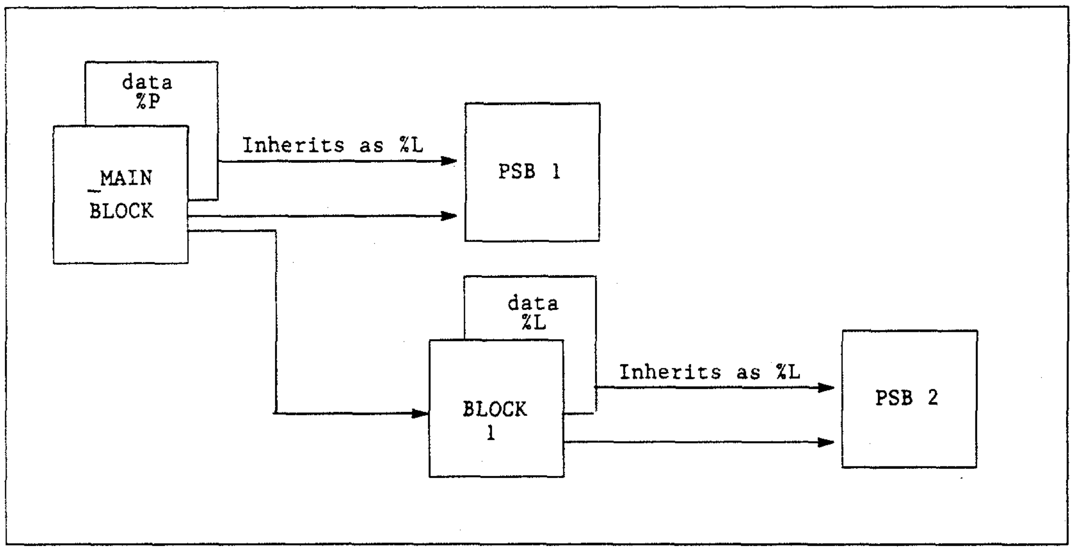
Parameterized Subroutine Blocks

A Parameterized Subroutine Block (PSB) is an optional user-defined function block, configured with between zero and seven input/output parameter pairs.

As with other blocks, Parameterized Subroutine Blocks can be called by the _MAIN block, other blocks, or itself. The calling block may pass parameters to the Parameterized Subroutine Block. When a Parameterized Subroutine Block is declared, it must be assigned a unique block name along with the number, type, and length of the parameters. Each parameter, other than the ENABLE and OK parameters, is designated as a BIT, WORD, or NWORD type, along with a specified length. BIT lengths range from 1 to 256; WORD and NWORD lengths range from 1 to 512. Default is one bit for BIT lengths or one word for WORD and NWORD lengths. In addition, you may also declare an optional three-character formal parameter reference name. For more information on defining and using Parameterized Subroutine Blocks and their parameters, refer to the "Parameterized Subroutines" section of chapter 3 in the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263.

Parameterized Subroutine Blocks and Local Data

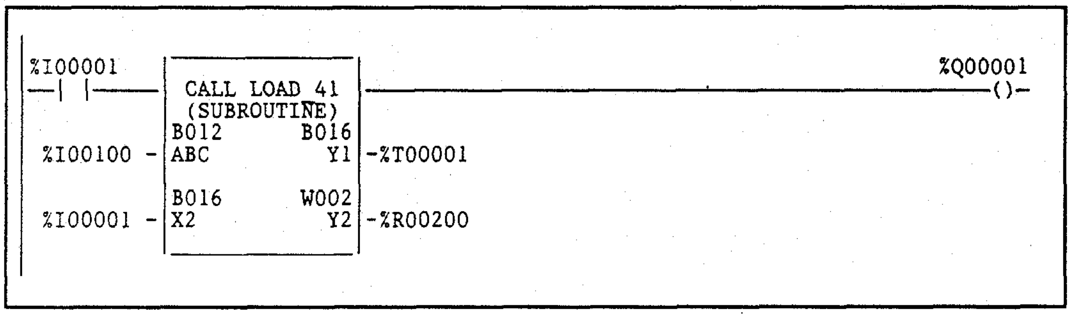
Parameterized subroutine blocks support the use of %P global data. Parameterized subroutine blocks do not have their own %L data, but instead they inherit the %L data of the calling block. Parameterized subroutine blocks also inherit %S contacts, such as FST_EXE, from the calling block. If %L references are used within a Parameterized Subroutine Block and the block is called by _MAIN, %L references will be inherited from the %P references wherever encountered in the Parameterized Subroutine Block (e.g., %L0005 = %P0005).



How Parameterized Subroutine Blocks Are Called

A Parameterized Subroutine Block executes when called from the program logic in the _MAIN block, another block, or itself.

In the following example, if %I00001 is set, the parameterized subroutine named LOAD_41 is executed. The LOAD_41 subroutine block operates on the input data (located at reference addresses %I00100 – %I00111 and %I00001 – %I00016) and produces values in the block of output data (located at reference addresses %T00001 – %T00016, and at register memory addresses %R00200 – %R00201). The logic within the subroutine can also control the OK output of the Parameterized Subroutine Block.



This example shows the subroutine CALL instruction as it will appear in the calling block. By positioning the cursor within the instruction, you can press **F10** to zoom into the parameterized subroutine.

For more information on the CALL SUBROUTINE instruction, refer to page 4-170 of this manual.

Referencing Formal Parameters within a Parameterized Subroutine Block

Formal parameters are those parameters used within the Parameterized Subroutine Block that are passed from and to the calling block. They are either BIT, WORD, or NWORD types. NWORD type parameters may be used on any multi-word type operands, but not on discrete types. (An NWORD is a number of words passed into a Parameterized Subroutine Block. See the "Formal Parameters within a Parameterized Subroutine Block" section of section 9, "Parameterized Subroutine," in chapter 3, of the *Logicmaster™ 90-70 Programming Software User's Manual* (GFK-0263) for a complete definition and guidelines for usage of NWORDs.)

The formal parameters are identified as **X** input parameters or **Y** output parameters, followed by the number of the input or output parameter, respectively. For example, X2 indicates the parameter used at location X2 in the parameterized subroutine declaration. The X2 label could be followed by a value of 1 to 16 to the length provided in the subroutine declaration (B016).

Up to seven formal parameter pairs may be declared in a Parameterized Subroutine Block. The formal parameter type, number, and length use the form:

```
ab[ccc]
```

```
where:  a = X     denotes an input formal parameter.
        a = Y     denotes an output formal parameter.
        b         is a parameter number between 1 and 7.
        c         is a valid BIT, WORD, or NWORD index.
```

The labels X1 through X7 and Y1 through Y7 may be assigned a nickname of up to three characters. Refer to the *Logicmaster 90-70 Programming Software User's Manual*, GFK-0263, for more information on assigning a nickname.

Assigned parameters are PLC references or data flow which pass their address or data into or out of a Parameterized Subroutine Block. An assigned parameter may pass either the value of the data in the assigned parameter (BIT type parameters) or the address of the assigned parameter (WORD or NWORD type parameters). Assigned parameters are defined in a parameter assignment table, which can be accessed by zooming into any Parameterized Subroutine Block and pressing **ALT-W**.

Restrictions on Formal Parameters within a Parameterized Subroutine Block

In general, formal BIT parameters are allowed on all contacts, coils, and function block parameters that allow discrete references (%I, %Q, %M, %T, %S, %G, and %U). Formal WORD and NWORD parameters are allowed on all function block parameters that allow register references (%R, %AI, %AQ, %P, %L, and %UR). NWORD parameters are only allowed on multi-word type parameters (i.e., DINT, DWORD, or REAL). (An NWORD is a number of words passed into a Parameterized Subroutine Block. See the "Formal Parameters within a Parameterized Subroutine Block" section of section 9, "Parameterized Subroutine," in chapter 3, of the *Logicmaster™ 90-70 Programming Software User's Manual* (GFK-0263) for a complete definition and guidelines for usage of NWORDs.) The following list contains several exceptions and restrictions which have been identified when using formal parameters within a Parameterized Subroutine Block:

1. Transitional contacts, transitional coils, and retentive coils are not allowed with formal parameters. The editor will substitute the non-retentive equivalent of these functions, `+---(M)`, `+---(SM)`, and `+---(/M)`, and display an appropriate warning message.
2. Formal BIT input parameters cannot be used as output parameters on a function block.
3. The DOIO function is not allowed with formal parameters.
4. Multi-word type function block (i.e., DINT, DWORD, or REAL) parameters are only allowed with formal NWORD parameters.
5. Formal parameters are not allowed on the following function block parameters:

Function	Parameter
Service Request (SVCREQ)	PARMS input parameter.
Communications Request (COMMREQ)	IN input parameter.
DATA_INIT DATA_INIT_COMM DATA_INIT_PID DATA_INIT_ASCII	Q output parameter.

6. A Parameterized Subroutine Block's BIT type formal parameters may not be passed to another Parameterized Subroutine Block.
7. WORD formal parameters cannot be passed into another Parameterized Subroutine Block's NWORD input parameter.

External Blocks

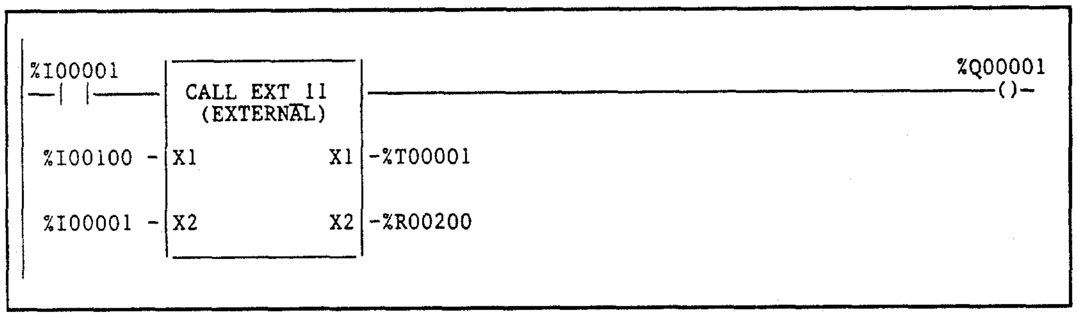
External blocks are created using the C Programmer's Toolkit. Refer to the C *Programmer's Toolkit for Series 90-70 PLCs* (GFK-0646C) for detailed information regarding external blocks.

How External Blocks Are Called

External blocks are added to a user program by using the Librarian function in the Logicmaster software package. For information regarding importing external program blocks, refer to the "Program Blocks, External Blocks, and Interrupts" section of chapter 3, "Program Editing," of the *Logicmaster™ 90-70 Programming Software User's Manual,* GFK-0263.

An external block executes when called from the program logic in the _MAIN block or from another block. To facilitate the passing and returning of data, an external block may have 0 to 7 parameter pairs.

In the following example, if %I00001 is set, the external block named EXT_11 is executed. The block operates on the input data, located at reference addresses %I00100 – %I00111 and %I00001 – %I00016, and produces values in the block of output data, located at reference addresses %T00001 – %T00016, and at register memory addresses %R00200 – %R00201. The logic within the block can also control the OK output of the external block.

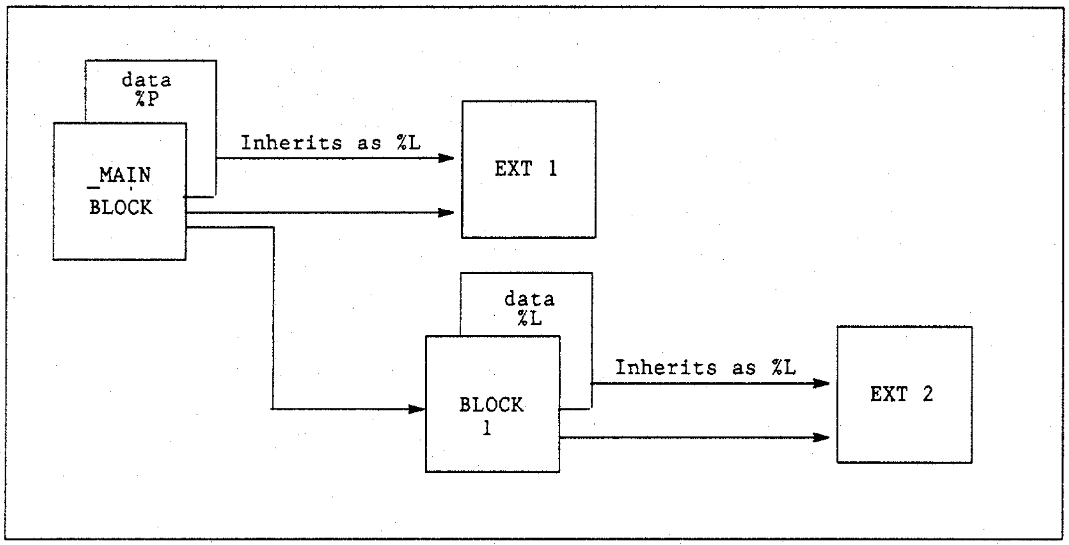


Note

Unlike other block types, external blocks cannot call any other blocks.

External Blocks and Local Data

External blocks support the use of %P global data. External blocks do not have their own %L data, but instead they inherit the %L data of the calling block. External blocks also inherit %S contacts, such as FST_EXE, from the calling block. If %L references are used within a external block and the block is called by _MAIN, %L references will be inherited from the %P references wherever encountered in the external block (e.g., %L0005 = %P0005).



Local Data Initialization

When an external block is stored to the PLC, a copy of its internal data is saved off. This data is used to re-initialize the block's data area whenever the PLC transitions from **STOP** to **RUN**.

External/standalone programs do not use %L data, but the internal data they use is somewhat similar to local data as discussed above. Do not confuse internal data used in a standalone program with %L data.

Standalone C Programs

Like external blocks, standalone programs are developed using the C Programmer's Toolkit. Unlike external blocks, however, standalone C Programs can be up to 512K bytes in size. Standalone C programs cannot call other standalone programs, nor can they call any blocks within an RLD program. Similarly, blocks within an RLD program cannot call a standalone program. Instead, standalone C programs are scheduled for execution using one of several possible program scheduling modes, described on page 2-52.

A maximum of 16 standalone programs can be used at one time. If an RLD program is used, only 15 standalone programs are allowed, for a total of 16 programs.

Note

Since standalone C programs are truly separate programs, they do not have access to memory types local to an RLD program (%L and %P). Do not confuse the internal or local data used in a standalone C program with %L and %P data.

Data Encapsulation

Each standalone C program is provided with a means of obtaining its own local copy of user data references. Instead of operating on the global set of user references, each standalone program can operate on its own local set of data. This feature is supported through the use of an *input/output specification*. For instructions on how to set the input/output specification for a program, refer to the "Specifying Local Data References" part of Chapter 3, "Program Editing," Section 9, "Multiple Programs: Program Declarations," in the *Logicmaster™ 90-70 Programming Software User's Manual*, GFK-0263.

The following steps occur when using an I/O specification with a standalone C program:

1. When the program is scheduled for execution, any corresponding input specification is copied from the global user reference data area(s) to an area local to the program.
2. As the program executes, it can operate on its local set of input and output data. Any interruptions during the execution phase will not affect this program's local copies of input or output data.
3. When the program completes, its local output specification is copied back to the specified set of user data reference areas.

Two particular concerns are addressed by using an input/output specification. First, if a program is suspended mid-execution and an output scan is performed before execution is resumed, output values will remain consistent since the output scan values are obtained from the user data reference locations. Second, if a program is interrupted mid-execution by another program, the first program is unaffected by changes to global data caused by the second program, since it has its own local copy of data.

Another benefit to using an input specification is to provide a more accurate sampling of input values. If a program's execution is postponed due to higher priority programs, the input specification may provide the program with a set of data that more accurately represents the state of that data when the program was scheduled since global user reference data may have been modified by higher priority programs or by the scanning of input values.

Input/Output Specifications

Unlike external blocks, standalone programs cannot have any input or output parameters, neither can RLD programs. Standalone C programs may utilize an input/output specification which lists a maximum of eight input and eight output ranges. The input ranges will be copied to the program at the start of program execution. The output ranges are copied from the standalone C program to the global reference on the completion of program execution. Note that this differs from external block parameters which are passed by reference, not by value. This operation is especially important for programs which may be time-sliced over multiple sweeps, which can occur when using Microcycle Sweep mode[1].

Caution

When the PLC runs in Microcycle mode, programs can be suspended in the middle of execution, possibly in the middle of a line of C code or in the middle of a rung of logic or function block. If the program uses global references such as %Q, %R, etc., a possible inconsistent set of reference values may be present at the time an interrupt program or output scan occurs. This inconsistency could even be within a given reference value if the value is not accessed according to its type.

1. Incoherent data can result if a program uses global data and is suspended across multiple sweeps. The data referenced will be from two successive sweeps. Although data cannot be incoherent within a byte or word, global data should only be accessed using its basic type (byte, word, etc.), otherwise incoherency can apply to individual elements as well.

To illustrate the possibility of inconsistent output data, consider the following example program which updates a given output value in several locations. For this example, assume that the user has configured %AQ1 as an output specification from the second program, and that user reference %AQ1 contains the value 0 when the two programs begin execution.

Figure 2-3. Differences Between Accessing Global and Local (Internal) Data

Program Accessing Global Data	
<program begins>	
%AQ1 = 0	An output scan occurring after this line of code would output a 0 to AQ1.
%AQ1 = %AQ1 + 2	An output scan occurring after this line of code would output a 2 to AQ1.
%AQ1 = %AQ1 + 12	An output scan occurring after this line of code would output a 14 to AQ1.
%AQ1 = %AQ1 / 2	An output scan occurring after this line of code would output a 7 to AQ1.
<program completes>	
Further output scans will output a 7 to AQ1.	

Program Using I/O Specification	
<program begins>	
LOC[1] = 0	An output scan occurring after this line of code would output a 0 to AQ1.
LOC[1] = LOC[1]+2	An output scan occurring after this line of code would output a 0 to AQ1.
LOC[1]=LOC[1] + 12	An output scan occurring after this line of code would output a 0 to AQ1.
LOC[1]=LOC[1]/2	An output scan occurring after this line of code would output a 0 to AQ1.
<program completes> The PLC copies the output specification from a local area back to the global areas when the program completes.	
Further output scans will output a 7 to AQ1.	

Note

Remember that standalone C programs are truly separate programs, they do not have access to memory types local to an RLD program (%L and %P). Do not confuse the internal or "local" data used in a standalone C program with %L and %P data.

Standalone C Programs and Local Data

Standalone C programs do not have a local data copy provided by the PLC. Similarly they are not able to access %P memory of an RLD program, nor can they access any %L memory associated with a block within an RLD program. Standalone C programs do have local data that is declared within the C source file(s) used to create the standalone C program. Refer to the *C Programmer's Toolkit User's Manual* (GFK-0646) for further information.

Local Data Initialization

When a standalone C program is stored to the PLC, a copy of its internal data is saved off. This data is used to re-initialize the program's data area whenever the PLC transitions from **STOP** to **RUN**.

Referencing I/O Specification Data within a Standalone C Program

Several new C macros are defined for standalone C programs which are used to define and access input and output specification data. Refer to the C *Programmer's Toolkit for Series 90-70 PLCs* (GFK-0646, revision C or later) for information regarding referencing I/O specification data within a standalone C program.

Data Coherency of I/O Specifications

Since standalone C programs can be interrupted by other programs and interrupt blocks, data incoherency within an I/O specification can occur. Each individual I/O specification is limited to 2048 bytes, for a maximum of 16k bytes of input data and 16k bytes of output data. The 90-70 PLC will ensure the following:

- Each byte within an individual I/O specification is coherent with respect to that individual specification.
- If the total length of all input specifications is no more than 2048 bytes, the entire input specification will be coherent.
- If the total length of all output specifications is no more than 2048 bytes, the entire output specification will be coherent.

If the total length of an input or output specification exceeds 2048 bytes, groups of individual specifications whose combined lengths do not exceed 2048 bytes will be coherent. The following table indicates coherency in this case.

Table 2-11. Coherency of I/O Specification

I/O specification	Length (bytes)
1	2000
2	48
Interrupts may occur	
3	1024
Interrupts may occur	
4	1026
Interrupts may occur	
5	10
6	20
7	20
8	20

Using RLD v. Standalone C Programs

Several options need to be considered when determining which type of program is to be used. These issues are discussed in greater detail in following sections. The following list summarizes many of the features supported in each of the types of programs:

- Interrupt Blocks are preferred over standalone programs when interrupt latency is a concern. The overhead to process an interrupt standalone program is much larger than that of an interrupt block.
- RLD programs may operate only on global user reference data. This can introduce data coherency problems when the RLD program is run in Microcycle mode and the program is suspended over multiple sweeps.
- Standalone C programs incur an 8 KB overhead per program.
- An RLD program is preferred over a standalone C program when using large amounts of boolean instructions. The RLD program is much better suited for relay-type logic than are standalone C programs.

Table 2-12. RLD vs. Standalone C Program Trade-offs

	Programming Language	Program Size Limit	Data Types Accessible	Local Data Size	Scheduling Modes	Data Encapsulation	Block Types Supported
RLD	Ladder Logic	smaller of 512k or available memory size, organized into 16k blocks.	All	8k %P, 8k %L per block	All*	No	LD SFC PSB External
Standalone	C, using the C programmers toolkit.	smaller of 512k or available memory size	All except %P, %L	unlimited, counts as part of program size	All *	Yes	n/a

*Using Microcycle Sweep mode with an RLD program is not recommended. RLD programs always operate on global data directly, which can lead to inconsistent output values if the RLD program is suspended mid-execution. See the section on Microcycle Sweep mode on page 2-49 for further details.

Differences in Operation: RLD and Standalone C Programs

Retentiveness of Data

When only standalone C programs are used, the retentive nature of data is based solely on memory type since there are no coil instructions. In this case %Q and %M are retentive. If both RLD and standalone C programs are used, the retentive property of memory types is driven by their use in the RLD program. For more information about retention of logic and data, refer to the "Retentiveness of Logic and Data" section on page 2-14. For more information on retentive properties of specific memory types, refer to the table on page 2-11.

Global Data

RLD programs only have access to global data areas since they do not have the ability to use Input/Output Specifications. This can lead to inconsistent output values if an RLD program is used in Microcycle Sweep mode.

Interrupt Execution

Interrupt blocks within the RLD program have the highest priority in the system. In addition, they cannot be preempted while standalone C programs can be.

Queuing of Interrupts

The 90-70 PLC can queue the invoking of interrupt blocks within the RLD program. A standalone C program triggered by an interrupt will not be queued should another interrupt occur during the processing of the first interrupt. In this case a fault will be logged in the PLC fault table.

System Status References

The following differences exist when using System Status References* (called "Convenience References" in previous editions of this manual):

- The reference FST_EXE is not available to standalone C programs.
- The reference FST_SCN does not refer to %S0001 within standalone C programs. Instead, a macro is provided by the C toolkit to provide identical functionality.

*For information on System Status References, refer to page 2-18 through 2-21.

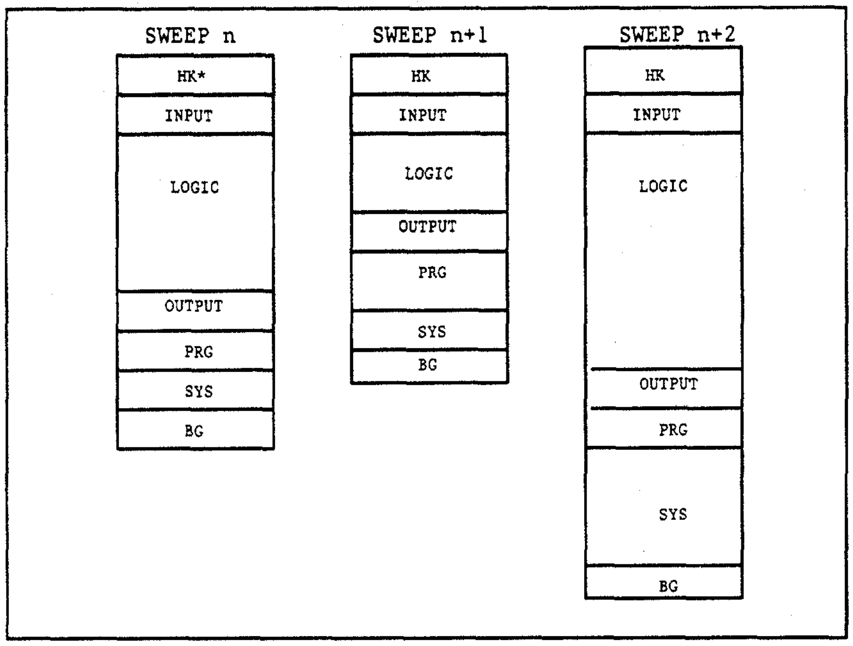
Section 4: *PLC Sweep Modes and Program Scheduling Modes*

Normal Sweep Mode

In Normal Sweep mode, each PLC sweep can consume a variable amount of time. The Logic Window is executed in its entirety each sweep. The Communications and Background Windows can be set to execute in a LIMITED or RUN-to-COMPLETION mode. Normal Sweep is the most common sweep mode used for PLC applications.

The following figure illustrates three successive PLC sweeps in Normal Sweep mode. Note that the total sweep times may vary due to sweep-to-sweep variations in the Logic Window, Communications Windows, and Background Window.

Figure 2-4. Typical Sweeps in Normal Sweep Mode



*The following abbreviations are used in the above figure: HK for Housekeeping PRG for Programmer Window, SYS for System Communications Window, BG for Background Window.

For more information on Normal Sweep mode, refer to chapter 5, "PLC Control and Status," and chapter 11, section 2, "Configuring the CPU Module," in the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263.

Constant Sweep Mode

In Constant Sweep mode, each PLC sweep begins at a specified Constant Sweep time after the previous PLC sweep began. The Logic Window is executed in its entirety each sweep. If there is sufficient time at the end of the sweep, the PLC will alternate among the Communications and Background Windows, allowing them to execute in RUN-to-COMPLETION mode until it is time for the next sweep to begin. Some or all of the Communications and Background Windows may not be executed. The Communications and Background Windows will terminate when the overall PLC sweep time has reached the value specified as the Constant Sweep time.

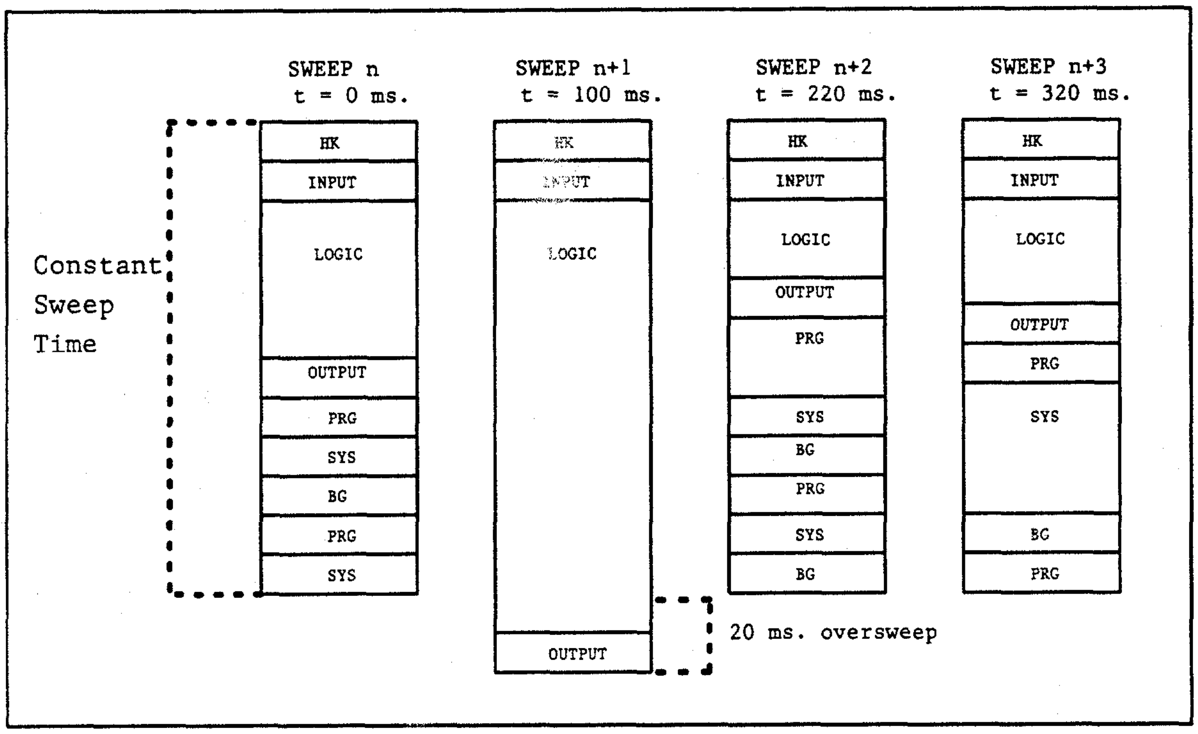
One reason for using Constant Sweep mode is to ensure that I/O are updated at constant intervals.

The value of the Constant Sweep timer can be configured using the Logicmaster 90-70 Configuration software and can be any value from 5 to 2550 milliseconds. The Constant Sweep timer value may also be set and Constant Sweep mode may be enabled or disabled by the Logicmaster 90-70 Programming software or by the user program using SVCREQ function #1. The Constant Sweep timer has no default value; a timer value must be set prior to or at the same time Constant Sweep mode is enabled.

If the PLC sweep exceeds the Constant Sweep time in a given sweep, the PLC places an oversweep alarm in the PLC fault table and sets the OV_SWP (%SA0002) status reference at the beginning of the next sweep. The OV_SWP status reference is reset when the time of the last sweep did not exceed the Constant Sweep timer or the PLC is not in Constant Sweep mode. Additional sweep time due to an oversweep condition in a given sweep does not affect the time given to the next sweep.

The following figure illustrates four successive PLC sweeps in Constant Sweep mode with a Constant Sweep time of 100 milliseconds. Note that the total sweep time is constant, but an oversweep may occur due to the Logic Window taking longer than normal.

Figure 2-5. Typical Sweeps in Constant Sweep Mode



*The following abbreviations are used in the above figure: HK for Housekeeping PRG for Programmer Window, SYS for System Communications Window, BG for Background Window.

For more information on Constant Sweep mode, refer to chapter 5, "PLC Control and Status," and chapter 11, section 2, "Configuring the CPU Module," in the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263.

Constant Window Mode

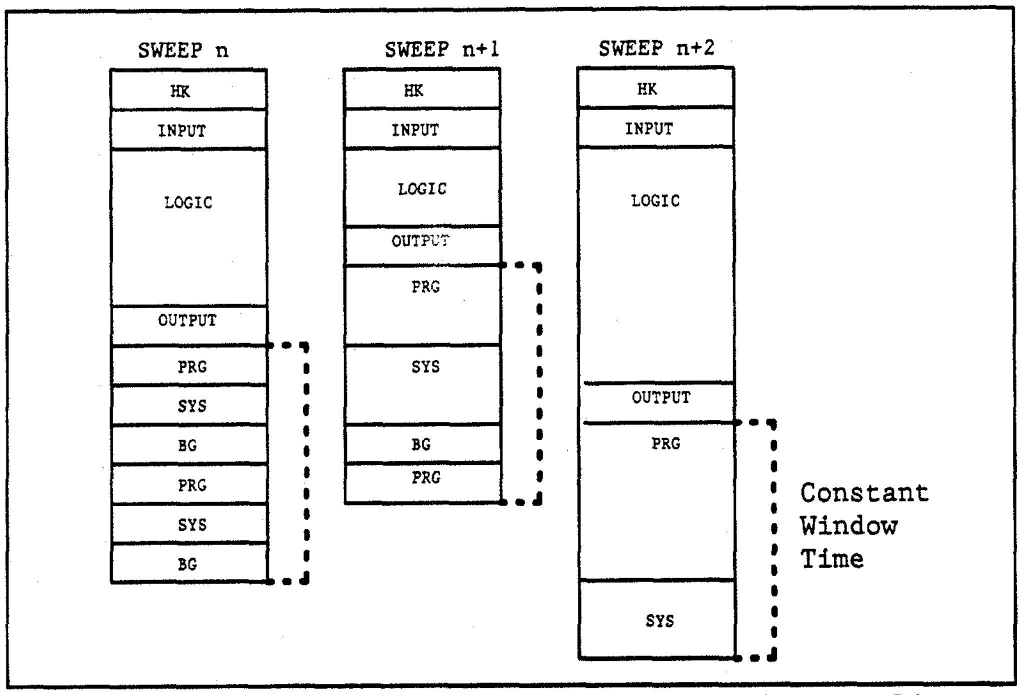
In Constant Window mode, each PLC sweep can consume a variable amount of time. The Logic Window is executed in its entirety each sweep. In this mode, the PLC will alternate among the three windows, allowing them to run in a RUN-to-COMPLETION mode for a time equal to the value set for the Constant Window timer. The overall PLC sweep time is equal to the time required to execute the Housekeeping, Input Scan, Logic Window, and Output Scan phases of the sweep plus the value of the Constant Window timer. This time may vary due to sweep-to-sweep variances in the execution time of the Logic Window.

An application which requires a certain amount of time between the Output Scan and the Input Scan, permitting inputs to settle after receiving output data from the program, would be ideal for Constant Window mode.

The value of the Constant Window timer can be configured using the Logicmaster 90-70 Configuration software and can be any value from 5 to 255 milliseconds. The Constant Window timer value may also be set by the Logicmaster 90-70 Programming software or by the user program using SVCREQ functions #3, #4, and #5.

The following figure illustrates three successive PLC sweeps in Constant Window mode. Note that the total sweep times may vary due to sweep-to-sweep variations in the Logic Window, but the time given to the Communications and Background Windows is constant. Some of the Communications or Background Windows may be skipped, suspended, or run multiple times based on the Constant Window time.

Figure 2-6. Typical Sweeps in Constant Window Mode



*The following abbreviations are used in the above figure: HK for Housekeeping PRG for Programmer Window, SYS for System Communications Window, BG for Background Window.

For more information on Constant Window mode, refer to chapter 5, "PLC Control and Status," and chapter 11, section 2, "Configuring the CPU Module," in the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263.

Microcycle Sweep Mode

In Microcycle Sweep mode, each PLC sweep begins at an absolute time, which is a multiple of the base cycle time, relative to the STOP-to-RUN transition of the PLC. The base cycle time specifies how long each sweep should take (similar to the Constant Sweep time in Constant Sweep mode). The user programs are scheduled for execution each sweep based on their period and may execute in a time-sliced fashion over multiple PLC sweeps. The PLC will alternate between the Communications and Background Windows, allowing them to run in a RUN-to-COMPLETION mode until it is time for the next sweep to begin.

Microcycle Sweep mode can be used to allow some programs to execute more often than others. This allows more processing time to be applied to the more important or more time-critical tasks. Microcycle Sweep mode also allows programs to execute more in line with the time when their inputs are available.

Although Microcycle Sweep mode has a fixed sweep time, it is significantly different from Constant Sweep mode. First, user programs do not necessarily execute in their entirety each sweep. In order to maintain the base cycle time and the Communications and Background Window times, user programs may be suspended during execution and resumed the following PLC sweep. Also, additional sweep time due to an oversweep condition in a given sweep causes the next sweep to be shortened by the oversweep time. In this way each PLC sweep (with the exception of sweeps which follow an oversweep condition) begins at an absolute time relative to the STOP-to-RUN transition of the PLC. Finally, the Communications and Background Windows are guaranteed to run for at least the specified Window time each sweep. The Logic Window will be suspended, if necessary, to guarantee that the Communications and Background Windows get to run for the specified Window time.

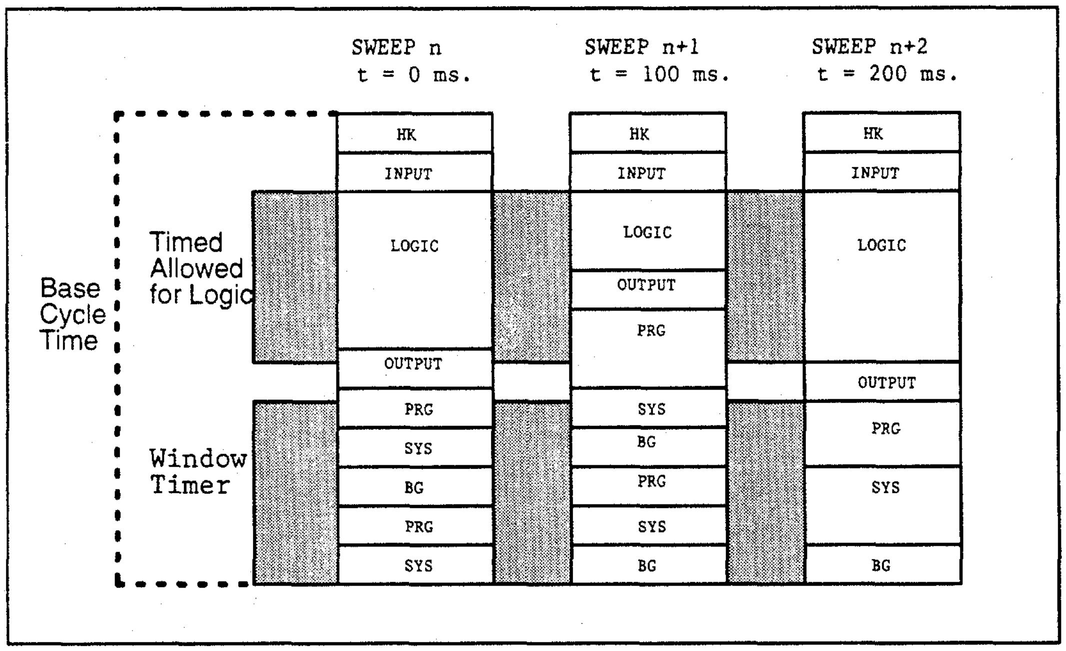
The base cycle time and the window timer value can be configured using the Logicmaster 90-70 Configuration software. The base cycle time can be any value from 5 to 2550 milliseconds. The Constant Window timer can be any value from 5 to 255 milliseconds. The base cycle time and Constant Window timer may also be set by the Logicmaster 90-70 Programming software while the PLC is in STOP mode. The base cycle time and window timer cannot be changed while the PLC is in RUN mode.

In Microcycle Sweep mode, Periodic programs execute on a priority basis. Periodic programs have priority inverse to their period (smallest period has highest priority). Refer to the next section titled "User Program Execution" for more information on Periodic programs and their execution.

If the PLC sweep exceeds the base cycle time in a given sweep, the PLC places an oversweep alarm in the PLC fault table and sets the OV_SWP (%SA0002) status reference at the beginning of the next sweep. The OV_SWP status reference is reset when last sweep time does not exceed the base cycle time. Sweep time due to an oversweep condition in a given sweep causes the next sweep to be shortened by the oversweep time.

The following figure illustrates three successive PLC sweeps in Microcycle Sweep mode with a base cycle time of 100 milliseconds. Note that the sweep time is constant and the Communications and Background Windows are guaranteed to run for the configured window timer. In sweep n and sweep n+1, the Logic Window finishes early and additional time is given to the Communications and Background Windows. In sweep n+2, the Logic Window is not complete and is suspended so that the Communications and Background Windows can run for the specified window time.

Figure 2-7. Typical Sweeps in Microcycle Sweep Mode



*The following abbreviations are used in the above figure: HK for Housekeeping PRG for Programmer Window, SYS for System Communications Window, BG for Background Window.

Note

Run Mode Store (described in the "Storing to PLC from Programmer" section of chapter 9 in the *Logicmaster™ 90-70 Programming Software User's Manual,* GFK-0263) of logic, including ALT-S, is not supported in Microcycle Sweep mode. Also, the Single Sweep Debug feature is not supported in Microcycle Sweep mode.

For more information on Microcycle Sweep mode, refer to chapter 5, "PLC Control and Status," and chapter 11, section 2, "Configuring the CPU Module," in the *Logicmaster™ 90-70 Programming Software User's Manual,* GFK-0263.

Program Scheduling Modes

Each user program in the 90-70 PLC can execute, subject to sweep mode restrictions, in one of four program scheduling modes. This section will briefly describe the following four available program scheduling modes:

- Ordered
- Timed
- Event-Triggered
- Periodic

Note

For instructions on entering or changing the scheduling modes, refer to section 9, "Multiple Programs: Program Declarations," in chapter 3 of the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263.

Ordered

Ordered programs are executed in the Logic Window with all other Ordered programs. Ordered programs are executed once per sweep in the sequence in which they are declared in the Program Declaration screen of the Logicmaster 90-70 Program Editor. (For instructions on using the Program Declaration screen, refer to section 9, "Multiple Programs: Program Declarations," in chapter 3 of the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263.) Ordered programs are not supported in Microcycle Sweep mode.

Timed

Timed programs are scheduled to execute on a specified time interval with an initial delay (if specified) applied on STOP-to-RUN transition of the PLC. Timed programs are scheduled to execute on a priority basis during any phase of the PLC sweep. Timed programs are not supported in Microcycle Sweep mode.

Event-Triggered

Event-Triggered programs are scheduled to execute on the receipt of a configured I/O interrupt. Event-Triggered programs are scheduled to execute on a priority basis during any phase of the PLC sweep when the PLC is in Normal Sweep, Constant Sweep, or Constant Window Sweep mode.

In Microcycle Sweep mode, Event-Triggered programs are scheduled to execute on a priority basis in the Logic Window. In this case, the execution of Event-Triggered programs may be time-sliced over multiple sweeps.

Periodic

Periodic programs are scheduled to execute in the Logic Window with all other Periodic programs. Periodic programs execute on a priority basis relative to all other programs and may be time-sliced over multiple sweeps. Periodic programs are only supported in Microcycle Sweep mode.

Choosing PLC Sweep Mode and Program Scheduling Mode

Table 2-13. indicates the availability of each program scheduling mode in each of the available PLC sweep modes.

Table 2-13. Available Program Scheduling Modes in each PLC Sweep Mode

Sweep Mode	Program Scheduling Mode				Interrupt Blocks
	Ordered	Timed	Event-Triggered	Periodic	
Normal	Yes	Yes	Yes	No	Yes
Constant	Yes	Yes	Yes	No	Yes
Constant Window	Yes	Yes	Yes	No	Yes
Microcycle	No	No	Yes *	Yes	Yes

* Executes in Logic Window only.

User Program Execution

User Program Priorities

The priority of a user program specifies its priority relative to other programs. Higher priority programs execute before lower priority programs. If two or more programs with the same priority are scheduled at the same time, the order of execution is undefined. Programs can be suspended in the middle of execution by higher priority programs and interrupt blocks.

Ordered programs all have the same priority and are executed in the order in which they are declared in the Program Declaration screen in Logicmaster 90-70. Ordered programs have lower priority than Timed programs, Event-Triggered programs, and interrupt blocks.

Periodic programs have priority inverse to their period (smallest period has highest priority). The order of execution of Periodic programs with the same period is undefined. Periodic programs have lower priority than Timed programs, Event-Triggered programs, and interrupt blocks.

Timed and Event-Triggered programs have higher priority than Ordered and Periodic programs. The priority of a Timed or Event-Triggered program specifies its priority relative to other Timed and Event-Triggered programs. The priority range 10–99 (10 being the highest priority) is reserved for Timed and Event-Triggered programs which can run during any phase of the PLC sweep (i.e., not restricted to running in the Logic Window). Timed and Event-Triggered programs operate this way when the PLC is running in Normal Sweep, Constant Sweep, or Constant Window mode. In Microcycle Sweep mode, Event-Triggered programs are executed in the Logic Window, and priorities of 100–109 (100 being the highest priority) are reserved for this mode.

Timed and I/O interrupt blocks have the highest priority of any user logic.

Table 2-14. Priority Values for Timed and Event-Triggered Programs

Scheduling Mode	Sweep Mode	
	Normal Sweep Constant Sweep Constant Window	Microcycle
Ordered *	Executed in order declared in the Logic-master 90-70 Program Declaration screen.	Not supported.
Periodic *	Not supported.	Smallest period has highest priority.
Event-Triggered	10−99	100−109
Timed	10−99	Not supported.

* Ordered and Periodic scheduling modes have lower priority than Timed and Event-Triggered scheduling modes.

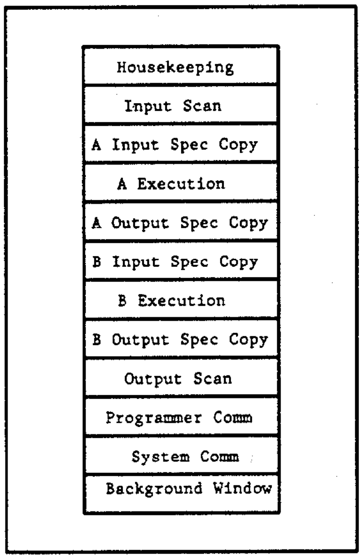
User Program Execution in Normal Sweep, Constant Sweep, and Constant Window Modes

In Normal Sweep, Constant Sweep, and Constant Window modes, the 90-70 PLC can execute Ordered, Timed, and Event-Triggered programs as well as Timed and I/O interrupt blocks.

Ordered programs execute in their entirety once per sweep in the Logic Window. The programs execute in the order in which they are declared in the Program Declaration screen in the Logicmaster 90-70 Program Editor. (For instructions on entering or changing the scheduling modes, refer to section 9, "Multiple Programs: Program Declarations," in chapter 3 of the *Logicmaster 90-70 Programming Software User's Manual*, GFK-0263.) The input specification is copied prior to execution of the program, and the output specification is copied upon completion of the program. In this way, the output of one program can be used as input for the next, if desired.

The following figure depicts two Ordered programs (A and B) executing in a typical PLC sweep in Normal Sweep mode.

Figure 2-8. Ordered Program Execution Sequence

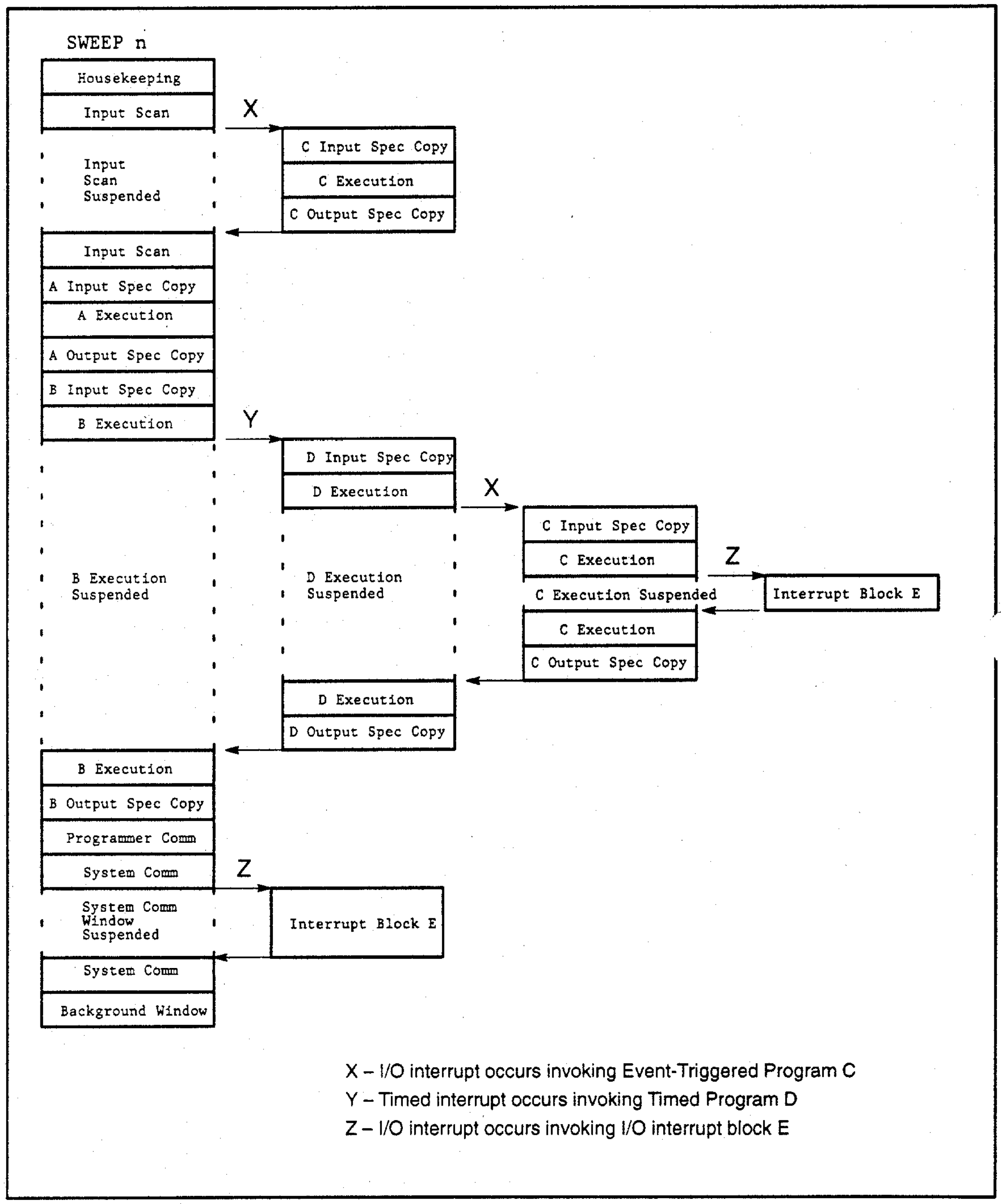


In Normal Sweep, Constant Sweep, and Constant Window mode, Timed and Event-Triggered programs execute during any phase of the PLC sweep. These programs will preempt the execution of Ordered programs and lower priority Timed and Event-Triggered programs. The input specification is copied at the time the program is scheduled to execute (i.e., when the time interval expires or the I/O interrupt occurs). The output specification is copied upon completion of the program.

Timed and I/O interrupt blocks execute during any phase of the PLC sweep. These blocks will preempt the execution of all programs and have the highest priority of any user logic in the PLC. Timed and I/O interrupt blocks do not have an input or output specification copy.

The following figure depicts 2 Ordered programs (A and B) , an Event-Triggered program (C) with priority 10, a Timed program (D) with priority 20, and an I/O interrupt block all executing in a typical PLC sweep in Normal Sweep mode.

Figure 2-9. Ordered, Timed, Event-Triggered and Interrupt Block Execution Sequence



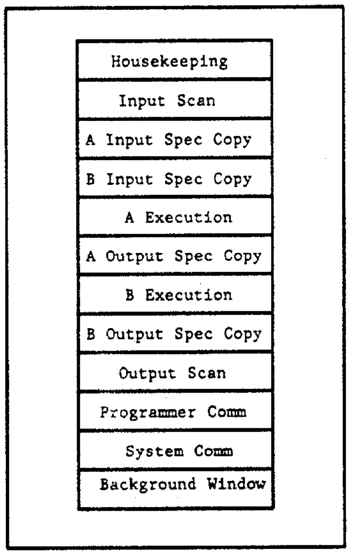
User Program Execution in Microcycle Sweep Mode

In Microcycle Sweep mode, the 90-70 PLC can execute Periodic and Event-Triggered programs as well as Timed and I/O interrupt blocks.

Periodic programs execute in the Logic Window. These programs are scheduled to execute based on the program's period. For example, a program with a period of 1 will be scheduled to execute every PLC sweep and a program with a period of 2 will be scheduled to execute every other PLC sweep. Periodic programs have priority inverse to their period (smallest period has highest priority). These programs are subject to time-sliced execution over multiple sweeps based on the time available to the Logic Window. Unlike Ordered programs, the input specification is copied at the beginning of the Logic Window for all Periodic programs which are scheduled to begin execution in a given sweep. In other words, all input specification copies will occur for Periodic programs before any of the Periodic programs begin, or continue, executing. The output specification is copied upon completion of the program.

The following figure depicts two (2) Periodic programs (A and B) executing in a typical PLC sweep in Microcycle Sweep mode.

Figure 2-10. Periodic Program Execution Sequence

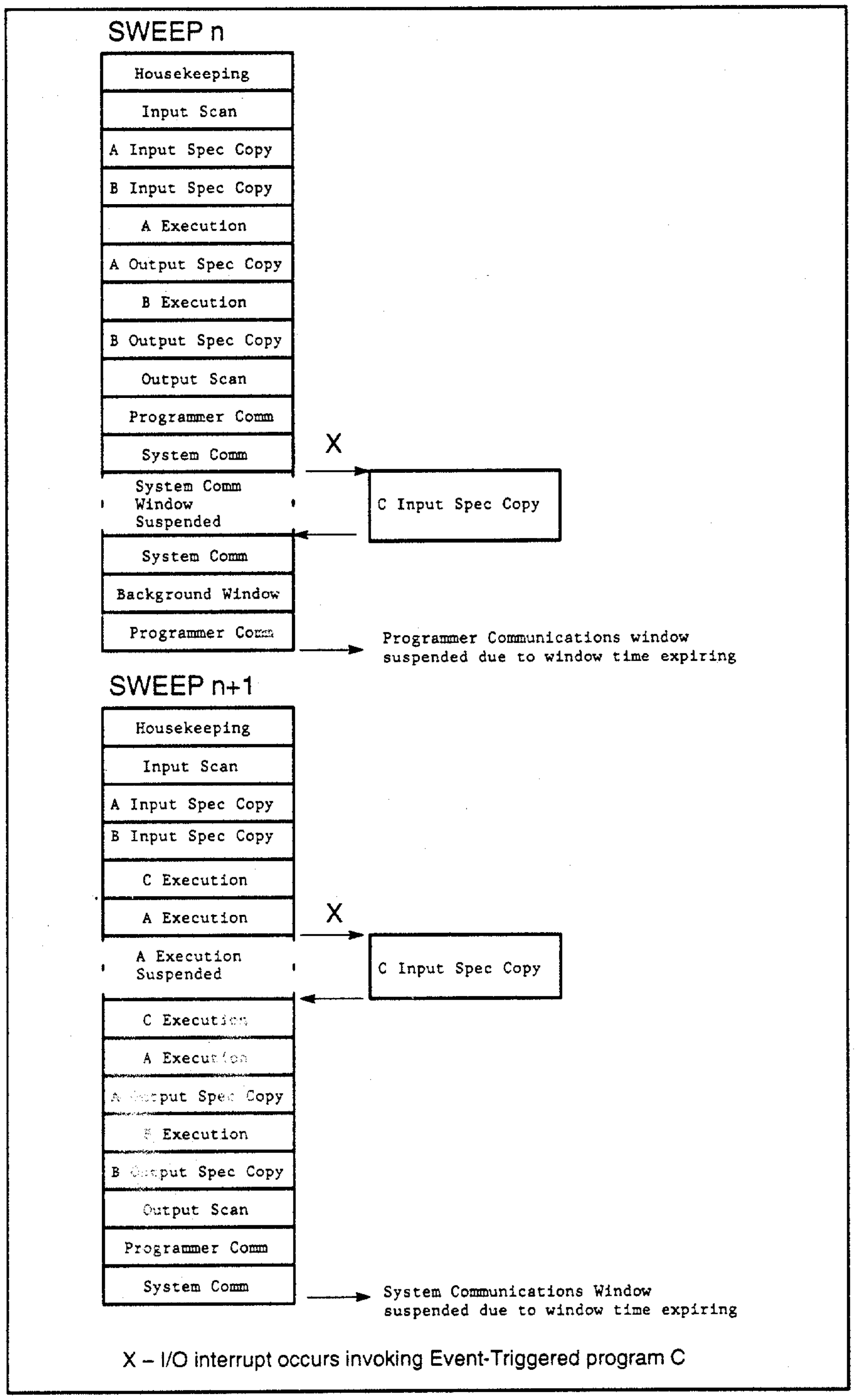


Unlike other sweep modes, Event-Triggered programs execute in the Logic Window only when the PLC is in Microcycle Sweep mode. If the I/O interrupt occurs during, or prior to, the end of the Logic Window, the Event-Triggered program will be scheduled to execute in the Logic Window of the current PLC sweep. Otherwise, it will be scheduled to execute in the Logic Window of the next PLC sweep. Event-Triggered programs will preempt the execution, or resumption, of Periodic programs and lower priority Event-Triggered programs. These programs are subject to the same time-sliced execution over multiple sweeps as Periodic programs, based on the time available to the Logic Window. The input specification for an Event-Triggered program is copied at the time the program is scheduled to execute (i.e., when the I/O interrupt occurs) not at the beginning of the Logic Window as with Periodic programs. The output specification is copied upon completion of the program.

Timed and I/O interrupt blocks execute during any phase of the PLC sweep when the PLC is in Microcycle Sweep mode. These blocks will preempt the execution of all programs and have the highest priority of any user logic in the PLC. Timed and I/O interrupt blocks do not have an input or output specification copy.

The following figure depicts two Periodic programs (A and B) and one Event-Triggered program (C) executing in two successive Microcycle Sweeps. Periodic programs A and B both have a period of 1.

Figure 2-11. Periodic and Event-Triggered Execution Sequence



Global Data in Microcycle Sweep Mode

Incoherent data can result if a program uses global data (%R, %I, %Q, etc.) and is suspended across multiple sweeps. The data referenced will be from two successive sweeps. Although data cannot be incoherent within a byte or word, global data should only be accessed using its basic type (byte, word, etc.); otherwise incoherency can apply to individual elements as well. If at all possible, the input and output specifications should be used to access and update global data areas.

Interrupt Handling

There are two types of interrupts available for user program handling in the 90-70 PLC.

I/O Interrupts These interrupts are generated by 90-70 I/O modules to indicate discrete input state changes (rising/falling edge), analog range limits (low/high alarms), and high speed signal counting events.

Timed Interrupts These interrupts are generated by the 90-70 PLC CPU based on a user specified time interval with an initial delay (if specified) applied on STOP-to-RUN transition of the PLC.

Both of these types of interrupts may invoke a user program or block.

Caution

Interrupt blocks and programs can interrupt the execution of non-interrupt logic as well as other TIMED and EVENT-TRIGGERED programs. Therefore, unexpected results may occur if the interrupting logic and interrupted logic access the same data. If necessary, SVCREQ #17 or SVCREQ # 32 can be used to temporarily mask I/O and timed interrupt blocks and programs from executing when shared data is being accessed.

Interrupt Handling and Scheduling with Blocks

An interrupt block has the highest priority of any user logic in the system, and may be programmed to execute upon the receipt of a timed or I/O interrupt. The execution of a block triggered from a timed or I/O interrupt preempts the execution of the normal PLC sweep activities. Execution of the normal PLC sweep activities is resumed after the interrupt block completes. There can be a maximum of 64 I/O interrupt blocks and 16 timed interrupts blocks.

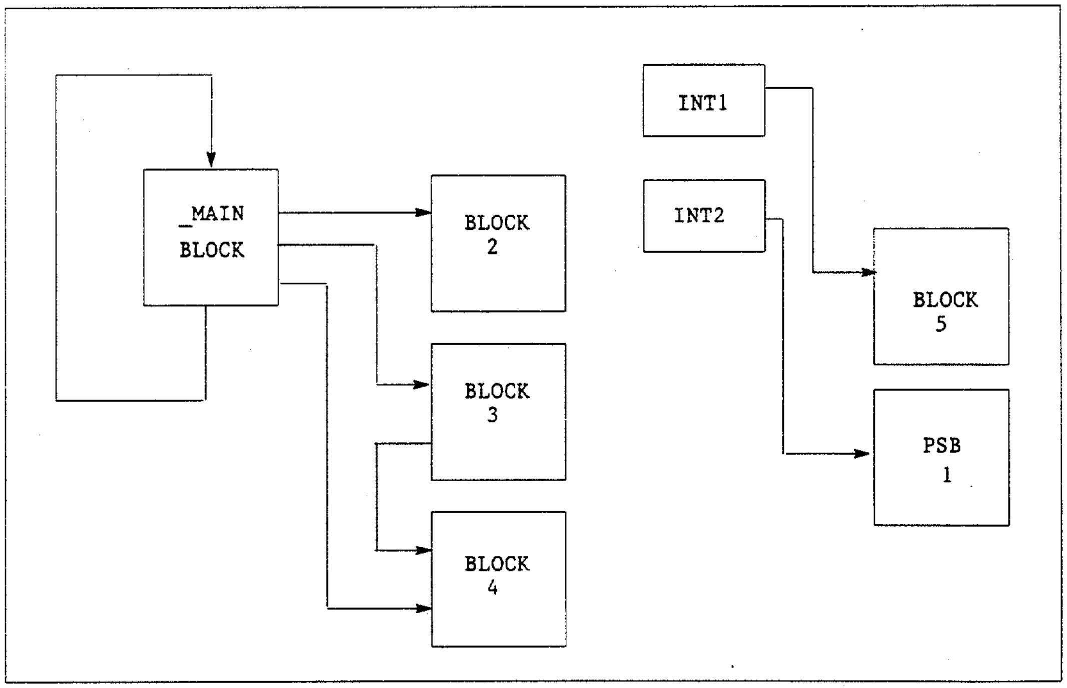
Note

Timer function blocks do not accumulate time if used in a block that is executed as a result of a timed or I/O interrupt.

Beginning with Release 6 of the PLC CPU, RLD interrupt blocks may make calls to other blocks. The application stack used during the execution of interrupt blocks is different from the stack used by the RLD program. Therefore, the nested call limit is different from the limit described for calls from the _MAIN block. The PLC will log an "Application Stack Overflow" fault and the PLC will transition to **STOP/FAULT** mode if a call results in insufficient stack space to complete the call.

Note

Blocks which may execute as a result of a timed or I/O interrupt should not be called from the _MAIN block or other non-interrupt blocks because portions of the code executed by blocks are not reentrant. In the example below INT1, INT2, BLOCK5, and PSB1 should not be called from _MAIN, BLOCK2, BLOCK3, or BLOCK4.



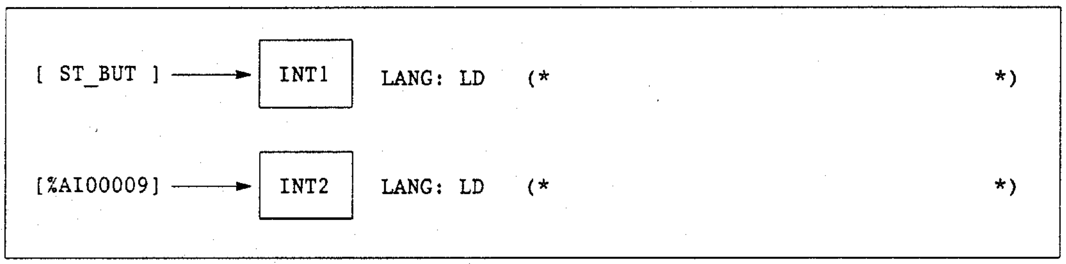
I/O Interrupt Blocks

A block may be triggered by an interrupt input from certain hardware modules. For example, on the 32-Circuit 24 VDC Input Module (IC697MDL650), the first input can be configured to generate an interrupt on either the rising or falling edge of the input signal. If the module is configured in this manner, that input can serve as a trigger to cause the execution of a RLD or External block.

To program an I/O interrupt block, the block must first be declared in the Logicmaster 90-70 block declaration section. It must then be associated with the interrupt through the use of an interrupt declaration.

The figure below shows two I/O interrupt declarations in the Logicmaster 90-70 interrupt declaration screen. The trigger ST_BUT calls RLD block INT1 if the input from a stop button wired to input 1 transitions in the configured direction. The module can be configured to generate the interrupt on a rising edge or a falling edge of the input. The RLD block INT2 is triggered by %AI00009.

Figure 2-12. I/O Interrupt Block Declarations



```
[ ST_BUT ] ———► INT1   LANG: LD   (*                                  *)

[%AI00009] ———► INT2   LANG: LD   (*                                  *)
```

Note

Parameterized subroutine blocks (PSBs) and External blocks (C blocks and C FBKs) with zero parameters may also be triggered by an interrupt input. For these types of blocks, the local data (%L) is inherited from the _MAIN local data (%P), for example, %L0005 = %P0005.

For more information on interrupt declarations, refer to chapter 3, section 8, "Blocks, External Blocks, and Interrupts," in the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263.

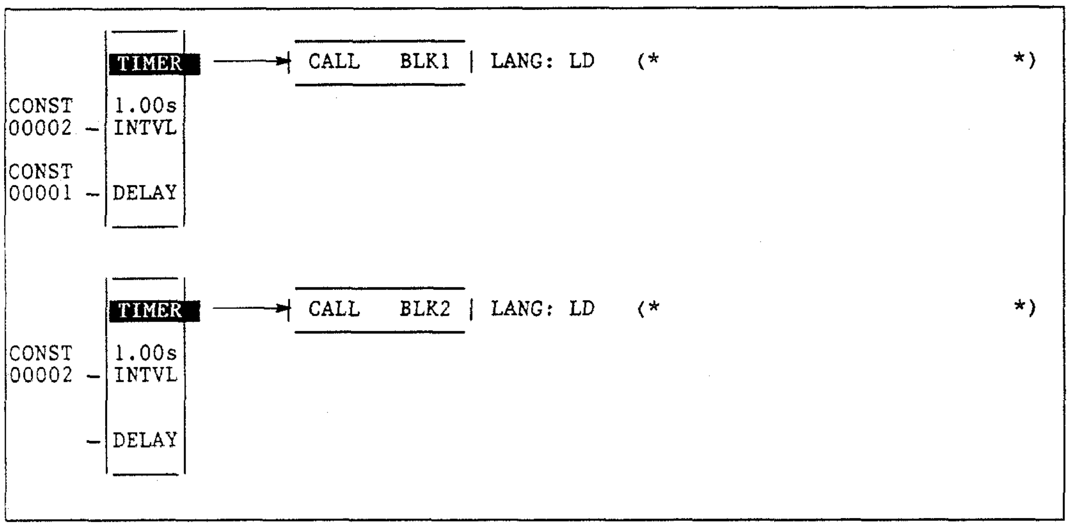
Timed Interrupt Blocks

A block may be executed on a user specified time interval with an initial delay (if specified) applied on STOP-to-RUN transition of the PLC. The time base options for timed interrupt blocks are 1.0s, 0.10s, 0.01s, and 0.001s.

To program a timed interrupt block, the block must first be declared in the Logicmaster 90-70 block declaration section. It must then be associated with a timed interrupt and given an interval and initial delay through the use of an interrupt declaration.

The first execution of a timed interrupt block will occur at ((DELAY * time base) + (INTVL * time base)) after the PLC is placed in **RUN** mode. The figure below shows two timed interrupt declarations in the Logicmaster 90-70 interrupt declaration screen. The RLD block BLK1 will be executed at times of 3 seconds, 5 seconds, 7 seconds, etc., after the PLC is placed in **RUN** mode. The RLD block BLK2 will be executed at two-second intervals, beginning two seconds after the PLC is placed in **RUN** mode. The absence of a DELAY value for BLK2 indicates that there will not be an initial delay in the first execution of the block.

Figure 2-13. Timed Interrupt Block Declarations



Parameters:

Parameter	Description
INTVL	INTVL is a constant value which will be multiplied by the time base of the interrupt to establish the frequency of execution of the associated block.
DELAY	DELAY is an optional field for the timed interrupt. It is a constant value which will be multiplied by the interrupt time base to establish an additional delay for the first execution of the associated block.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
INTVL															•	
DELAY															•	•

• = Valid data type, or place where power may flow through the function.

Note

Parameterized subroutine blocks (PSBs) and External blocks (C blocks and C FBKs) with zero parameters may also be triggered by a timed interrupt. For these types of blocks, the local data (%L) is inherited from the _MAIN local data (%P), for example, %L0005 = %P0005.

For more information on interrupt declarations, refer to chapter 3, section 8, "Blocks, External Blocks, and Interrupts," in the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263.

Interrupt Handling and Scheduling with User Programs

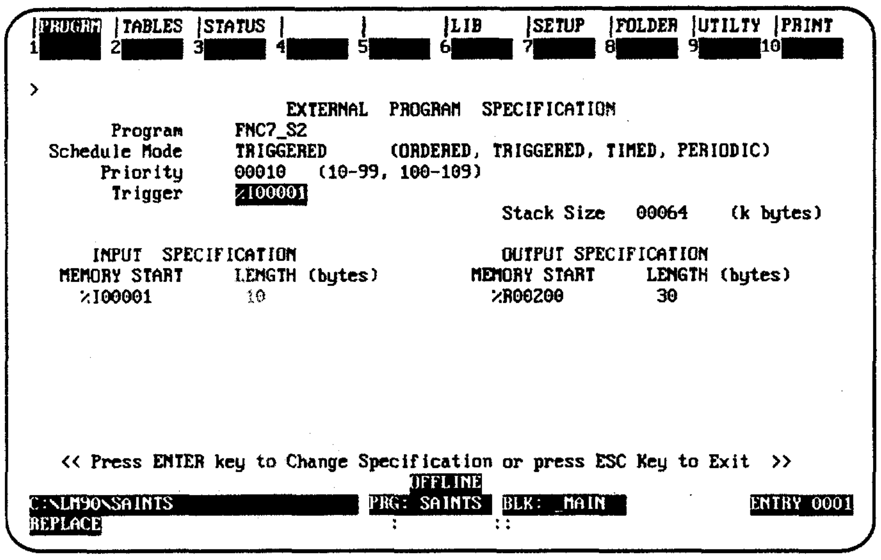
Event-Triggered Programs

Beginning with Release 6 of the PLC CPU, one of the scheduling modes available for user programs is activation of programs from an I/O interrupt. The Event-Triggered scheduling mode allows a user program to be invoked, along with its corresponding input and output specification copy, when a configured I/O interrupt occurs. Event-Triggered programs execute during any phase of the PLC sweep or only during the Logic Window, based on sweep mode. Refer to "User Program Execution" on page 2-53 for more information on the scheduling and execution of Event-Triggered programs.

To program an Event-Triggered program, the program must first be declared in the Logicmaster 90-70 Program Declaration screen. The scheduling mode must then be set to **TRIGGERED** using the Logicmaster 90-70 Program Specification screen. (For instructions on using the Program Declaration screen and Specification screen, refer to section 9, "Multiple Programs: Program Declarations," in chapter 3 of the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263.)

The following figure shows the program CPROG2 specified as a **TRIGGERED** program with a priority of 10. CPROG2 will execute at upon receipt of the configured I/O interrupt.

Figure 2-14. Event-Triggered Program Specification



```
|PROGRM |TABLES |STATUS |       |       |LIB    |SETUP  |FOLDER |UTILTY |PRINT
1       2       3       4       5       6       7       8       9       10

>
                        EXTERNAL  PROGRAM  SPECIFICATION
        Program      FNC7_S2
  Schedule Mode      TRIGGERED      (ORDERED, TRIGGERED, TIMED, PERIODIC)
       Priority      00010   (10-99, 100-109)
        Trigger      %I00001
                                                Stack Size   00064    (k bytes)

      INPUT  SPECIFICATION                      OUTPUT SPECIFICATION
   MEMORY START      LENGTH (bytes)           MEMORY START     LENGTH (bytes)
     %I00001          10                        %R00200          30




   << Press ENTER key to Change Specification or press ESC Key to Exit  >>
                                          OFFLINE
C:\LM90\SAINTS                           PRG: SAINTS  BLK: _MAIN            ENTRY 0001
REPLACE                                       :            ::
```

Timed Programs

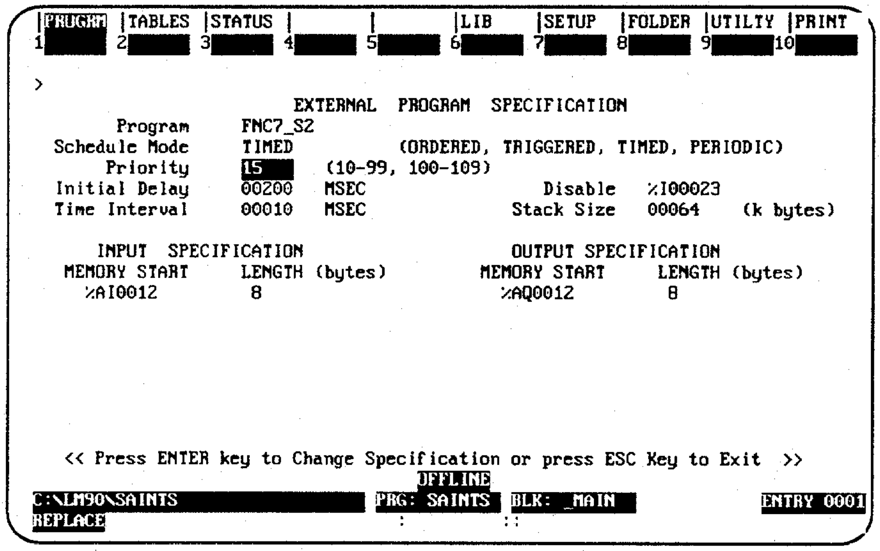
Beginning with Release 6 of the PLC CPU, one of the scheduling modes available for user programs is activation of a program from a timed interrupt. The Timed scheduling mode allows a user program to be executed, along with its corresponding input and output specification copy, on a user-specified time interval with an initial delay (if specified) applied on STOP-to-RUN transition of the PLC. Timed programs execute during any phase of the PLC sweep. (Refer to the "User Program Execution" section on page 2-53 for more information on the execution of Timed programs.)

To program a Timed program, the program must first be declared in the Logicmaster 90-70 Program Declaration screen. The scheduling mode must then be set to **TIMED** using the Logicmaster 90-70 Program Specification screen. (For instructions on using the Program Declaration screen and Specification screen, refer to section 9, "Multiple Programs: Program Declarations," in chapter 3 of the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263.)

The time base options for Timed programs are specified in milliseconds. The first execution of a Timed program will occur at [(Initial Delay) + (Time Interval * time base)] milliseconds after the PLC is placed in **RUN** mode.

In the following figure, the Timed program TMR2 will be executed at times of 210 ms, 220 ms, 230 ms, etc., after the PLC is placed in **RUN** mode. TMR2 has a priority of 15. TMR2 will execute at the specified time intervals unless the disable reference %I00023 is ON.

Figure 2-15. Timed Program Specification



```
|PROGRM |TABLES |STATUS |       |       |LIB    |SETUP  |FOLDER |UTILTY |PRINT
1       2       3       4       5       6       7       8       9       10

>
                     EXTERNAL  PROGRAM  SPECIFICATION
      Program      FNC7_S2
  Schedule Mode    TIMED          (ORDERED, TRIGGERED, TIMED, PERIODIC)
       Priority    15        (10-99, 100-109)
  Initial Delay    00200     MSEC                Disable   %I00023
  Time Interval    00010     MSEC             Stack Size   00064    (k bytes)

     INPUT  SPECIFICATION                   OUTPUT SPECIFICATION
   MEMORY START     LENGTH (bytes)        MEMORY START     LENGTH (bytes)
     %AI0012          8                     %AQ0012          8




    << Press ENTER key to Change Specification or press ESC Key to Exit  >>
                                          OFFLINE
C:\LM90\SAINTS                          PRG: SAINTS  BLK: _MAIN          ENTRY 0001
REPLACE
```

Note

Standalone programs execute on a priority basis. The user controls the setting of the priorities in **Event Triggered** and **Timed** scheduling modes via the Program Control screen in Logicmaster. For instructions on use of the Control screen, refer to chapter 3, section 9, "Multiple Programs: Program Declarations," in the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263.

Interrupt Blocks vs. Interrupt Programs

There are important differences that the user must be aware of when choosing a program instead of a block to handle an interrupt. When a block is selected to handle a Timed or I/O Interrupt, the block will execute immediately upon receipt of the interrupt and run until it completes. Interrupt blocks can execute during any phase of the PLC sweep, regardless of the current PLC sweep mode. Pending Timed interrupt blocks will execute before pending I/O interrupt blocks, but once an interrupt block (Timed or I/O) begins executing, it will run until it completes. If an interrupt occurs which attempts to execute a Timed or I/O interrupt block which has not fully completed execution due to a previous interrupt, the interrupt will be 'queued' and the block will be executed again after the interrupt block completes execution. If an interrupt block has already been 'queued' in this manner once, any additional interrupts which occur for this block will be ignored.

Upon receipt of the interrupt, Timed or Event-Triggered programs are immediately scheduled to begin execution (including the copying of the input specification). However, the actual execution of the program occurs on a priority basis. Unlike interrupt blocks, the execution of Timed or Event-Triggered programs can be delayed or preempted by other Timed or Event-Triggered programs of a higher priority as well as other interrupt blocks. Additionally, if an interrupt occurs which attempts to schedule a Timed or Event-Triggered program which has not fully completed execution due to a previous interrupt, a "Program not Readied" application fault will be logged in the PLC fault table and the interrupt will be ignored.

When the PLC is in Normal Sweep, Constant Sweep, or Constant Window mode, interrupt programs can execute during *any* phase of the PLC sweep. When the PLC is in Microcycle Sweep mode, interrupt programs are scheduled to execute in the Logic Window.

In summary, the primary differences between interrupt blocks and interrupt programs are as follows:

Interrupt Block

- Executed immediately upon receipt of interrupt.
- Cannot be preempted by other logic once interrupt block begins execution.
- One additional interrupt is "queued" if the block is still executing due to a previous interrupt.
- Executes during any phase of the PLC sweep.

Interrupt Program

- Scheduled to execute on a priority basis.
- Can be preempted by higher priority interrupt program or interrupt block.
- Additional interrupts are ignored and a fault is logged if the program is still executing due to a previous interrupt.
- Executes during any phase of the PLC sweep when PLC is in Normal Sweep, Constant Sweep, or Constant Window mode.
- Executes in Logic Window when PLC is in Microcycle Sweep mode.

Section 5: Run/Stop Operations

Modes of Operation

Four run/stop modes of operation are supported by the 90-70 PLC. You can change these modes in the following ways: the toggle switch, Logicmaster software, RLD function blocks, and system calls from C applications. Switching to and from various modes can be restricted based on privilege levels, position of the PLC toggle switch, passwords, etc.

Run/Outputs Enabled

In this mode, the PLC runs user programs and continually updates physical outputs, including Genius and Field Control outputs. The Programmer and System Communications Windows are run in either LIMITED, RUN-TO-COMPLETION, or CONSTANT mode.

Note

Storing %P and %L reference data in either type of Run mode can be problematic. Refer to the "Storing to the PLC from the Programmer" section of chapter 9 in the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263, for more information.

Run/Outputs Disabled

In this mode, the PLC runs user programs, but updates to physical outputs, including Genius and Field Control, are not performed. Physical outputs are held in their configured default state in this mode. The Programmer and System Communications Windows are run in either LIMITED, RUN-TO-COMPLETION, or CONSTANT mode.

Stop/IO Scan

In this mode the PLC does not run user programs, but the input and output scan phases of the PLC sweep are performed. The Programmer and System Communications Windows are run in RUN-TO-COMPLETION mode. The Background Window is LIMITED to 10 ms.

Note

Stop/IO Scan mode is **not** supported in Microcycle Sweep mode.

Stop/No IO Scan

In this mode the PLC does not run user programs, and the input or output scan phases of the PLC sweep are not performed. The Programmer and System Communications Windows are run in a RUN-TO-COMPLETION mode. The Background Window is LIMITED to 10 ms.

Mode Transitions

Stop-to-Run Transition

Several operations are performed by the CPU on Stop-to-Run transition. These operations include the following:

- Validation of sweep mode and program scheduling mode selections.
- Validation of references used by programs with the actual configured sizes.
- Re-initialization of data areas for external blocks and standalone C programs.
- Clearing of non-retentive memory.

Run-to-Stop Transition

Wind-Down Period for Microcycle Sweep Mode

When the PLC is running in Microcycle Sweep mode (refer to page 2-49 for information about Microcycle Sweep mode), a wind-down or logic solution period may occur after the PLC is commanded to Stop mode. This wind-down period is equal to the amount of time that the currently executing program(s) take to complete their execution unless that amount of time exceeds 2.5 seconds. If the currently executing programs exceed 2.5 seconds in their attempt to complete their executions, a fault will be logged in the PLC fault table, and the CPU will complete its transition to Stop mode. During the wind-down period, no additional programs (including interrupt programs and blocks) will be scheduled for execution. Input Scans, Output Scans, Communications Windows (Programmer and System), and the Background Window continue during the wind-down period.

Note

By definition, exceeding the CPU wind-down period means that not all programs completed execution prior to the PLC going to STOP mode. Furthermore, when the PLC is next commanded to RUN mode, all programs will begin execution at their normal beginning point. Program(s) are **not** resumed at their "wind-down exceeded" execution point.

Section 6: Power-Up and Power-Down Sequences

Power-Up

System power-up consists of the following parts:

- Power-up self-test.
- PLC memory validation
- System configuration
- Option module self-test completion
- Option module dual port interface tests
- I/O system initialization

Power-Up Self-Test

On system power-up, many modules in the system perform a power-up diagnostic self-test. Series 90-70 PLC modules execute hardware checks and software validity checks. Intelligent option modules perform setup and verification of on-board microprocessors, software checksum verification, local hardware verification, and notification to the CPU of self-check completion. Any failed tests are queued for reporting to the CPU during the system configuration portion of the cycle.

In the CPU, power-up will be either a quick power-up (a warm start) or a full power-up (a cold start), depending on whether the CPU is able to go to **RUN** mode after powering up. If all the conditions are met for the CPU to go to **RUN** mode—a valid program or configuration is present, the switch is in **RUN** mode, and no fatal fault exists—then the CPU will perform a quick power-up. If any of the conditions are not met, a full power-up is performed.

A quick power-up will only perform the CPU processor and BCP tests, along with a minimal RAM test. The goal of a quick power-up is to get the CPU up and running as quickly as possible. The remaining tests, ROM CRC, exhaustive memory tests, and peripheral tests are only performed on a full power-up.

If a low battery indication is present, then a low battery fault is logged into the PLC fault table.

PLC Memory Validation

The next phase of system power-up is the validation of the PLC memory within the CPU. First, the system verifies that the battery is not low and that battery-backed RAM areas are still valid. A known area of battery-backed application RAM is checked to determine if data was preserved. Next, if a ladder diagram program exists, then a checksum is calculated across the _MAIN ladder block. If no ladder diagram program exists, then a checksum is calculated across the smallest standalone C program.

When the system is sure that the application RAM is preserved, then a known area of the BCP bit cache area is checked to determine if the BCP bit cache data was preserved. If this test passes, then the Bit Cache memory is left containing its power-up values. (Non-retentive outputs are cleared on a transition from **STOP** to **RUN** mode.) If this checksum does not compare or the retentive test on the application RAM fails, the Bit Cache memory is assumed to be in error and all areas are cleared. The PLC is now in a cleared state, the same as if a new CPU module were installed. All logic and configuration files must be stored from the programmer to the PLC.

System Configuration

After completing its own self-test, the CPU performs the system configuration. It first clears all of the system diagnostic bits in the BCP Bit Cache memory. This prevents faults that were present before power-down, but are no longer present, from accidently remaining as faulted. Then it polls each module in the system, checking for completion of the module's self-test.

The CPU reads information from each module, comparing it with user-provided rack/slot configuration information. Any differences between actual configuration and user-specified configuration are logged in the fault tables.

Option Module Self-Test Completion

Option modules may take a longer time to complete their self-tests than the CPU due to the time required to test communications media or other interface devices. As an option module completes its initial self-tests, it tells the CPU the time required to complete the remainder of these self-tests. During this time, the CPU provides whatever additional information the module needs to complete its self-configuration, and the module continues self-tests and configuration. If the module does not report back in the time it specified, the CPU marks the module as faulted and makes an entry in one of the fault tables. When all self-tests are complete, the CPU obtains reports generated during the module's power-up self-test and places fault information (if any) in the fault tables.

Option Module Dual Port Interface Tests

After completion of the option module self-test and results reporting, integrity tests are jointly performed on the dual-port interface used by the CPU and option module for communications. These tests validate that the two modules are able to pass information back and forth, as well as verify the interrupt and semaphore capabilities needed by the communications protocol. After dual port interface tests are complete, the communications messaging system is initialized.

I/O System Initialization

If the module is a Model 70 input module, no further configuration is required. If the module is a Model 70 output module, the module is commanded to go to its default state. A Model 70 output module defaults to all inputs off at power-up and in failure mode, unless told otherwise. When the module is a Bus Transmitter Module (BTM), it is interrogated about what remote racks are present in the system. Based upon the BTM's response, the CPU adds those racks and their associated slots into the list of slots to be configured.

Finally, the I/O Scanner performs its initialization. It initializes all the I/O controllers in the system by establishing the I/O connections to each I/O bus on the I/O controller and obtaining all I/O configuration data from that I/O controller. This configuration data is compared with the user-specified I/O configuration and any differences reported in the I/O fault table. The I/O Scanner then sends each I/O controller a list of the I/O modules to be configured on the I/O bus. After the I/O controllers have been initialized, the I/O Scanner replaces the factory default settings in all I/O modules with any application-specified settings.

For Model 70 input modules, the board may be set to interrupt when the signal(s) change state, and whether the interrupt will occur when the signal(s) transitions from high to low or low to high. For Model 70 output modules, their default state may be changed from off to hold last state.

Power-Down Sequence

System power-down occurs when the power supply detects that incoming AC power has dropped for more than one power cycle. A signal line on the backplane is driven low to indicate the condition, which causes an interrupt to the CPU. From the time this signal occurs, a minimum of 5 milliseconds remain to complete power-down processing.

Retention of Data Memory Across Power Failure

Because application RAM and BCP memory are battery-backed, the following types of data are preserved across a power cycle:

- Application program
- Fault tables and other diagnostic data
- Checksums on programs and blocks
- Override data
- Data in register (%R), local register (%L), and program register (%P) memory
- Data in analog memory (%AI and %AQ)
- State of discrete inputs (%I)
- State of retentive discrete outputs (%Q)
- State of retentive discrete internals (%M)
- State of discrete system internals (system bits, fault bits, reserved bits)

The following types of data are not preserved across a power cycle:

- State of discrete temporary memory (%T)
- %M and %Q memories used on non-retentive -()- coils

Section 7: Clocks and Timers

Clocks and timers provided by the Series 90-70 PLC include an elapsed time clock, a time-of-day clock, and software and hardware watchdog timers. Three types of timer function blocks include an on-delay timer, an off-delay timer, and a start-reset timer. Timed contacts cycle on and off (in square-wave form) every 0.01 second, 0.1 second, 1.0 second, and 1 minute.

Elapsed Time Clock

The elapsed time clock uses 100 microsecond "ticks" to track the time elapsed since the CPU powered on. The clock is not retentive across a power failure; it restarts on each power-up. Once per second the hardware interrupts the CPU to enable a seconds count to be updated. This seconds count rolls over (seconds count returns to zero) approximately 100 years after the clock begins timing.

Because the elapsed time clock provides the base for system software operations and timer function blocks, it may not be reset from the user program or the programmer. However, the application program can read the current value of the elapsed time clock by using SVCREQ function #16, described in chapter 4, section 9, "Control Functions."

Time-of-Day Clock

The time of day in the Series 90-70 PLC is maintained by a hardware time-of-day clock. The time-of-day clock maintains the following seven time functions:

- Year (two digits)
- Month
- Day of month
- Hour
- Minute
- Second
- Day of week

The time-of-day clock is battery-backed and maintains its present state across a power failure. However, unless the user initializes the clock, the values it contains are meaningless. The application program can read and set the time-of-day clock using SVCREQ function #7. The time-of-day clock can also be read and set from the Logicmaster 90-70 Configuration software.

The time-of-day clock is designed to handle month-to-month and year-to-year transitions. It automatically compensates for leap years until the year 2099.

Watchdog Timer

Software Watchdog Timer

A software watchdog timer in the Series 90-70 PLC is designed to detect "failure to complete sweep" conditions. The timer value for the software watchdog timer is set by the user in the Logicmaster 90 configuration software. The allowable range for this timer is 10 to 2550 milliseconds; the default value is 200 milliseconds. The software watchdog timer always starts from zero at the beginning of each sweep.

The software watchdog timer is useful in detecting abnormal operation of the application program which prevents the PLC sweep from completing within the user-specified time. Examples of such abnormal application program conditions are as follows:

- Excessive recursive calling of a block
- Excessive looping (large loop count or large amounts of execution time for each iteration)
- Infinite execution loop

When selecting a software watchdog value, always set the value higher than the longest expected sweep time to prevent accidental expiration. For Constant Sweep and Microcycle Sweep modes, allowance for oversweep conditions should be considered when selecting the software watchdog timer value.

If the software watchdog timeout value is exceeded, the OK LED blinks, and the CPU goes to **STOP/HALT** mode. Certain functions, however, are still possible. A fault is placed in the PLC fault table, and outputs go to their default state. The CPU will only communicate with the programmer; no other communications or operations are possible. To recover, power must be cycled on the rack containing the CPU.

To extend the current sweep beyond the software watchdog timer value, the application program may restart the software watchdog timer using SVCREQ function #8. However, the software watchdog timer value may only be changed from the configuration software.

Hardware Watchdog Timer

A backup circuit provides additional protection for the PLC. If this backup circuit activates, the PLC is immediately placed in **RESET** mode. Outputs go to their default state; no communications of any form are possible, and the CPU will halt. To recover, power must be cycled.

Section 8: System Security

The Series 90-70 PLC supports the following three types of system security:

1. Passwords/privilege levels
2. OEM protection
3. Write protect keyswitch

Passwords and Privilege Levels

Passwords are a configurable feature of the Series 90-70 PLC. Their use is optional and may be set up using the configuration software. The purpose of passwords is to provide different levels of access privilege for the PLC when the programmer is in **ONLINE** or **MONITOR** mode. Passwords are not used if the programmer is in **OFFLINE** mode. The use of passwords may restrict:

- Changing I/O and PLC configuration data.
- Changing programs.
- Reading PLC data.
- Reading programs.
- Locking blocks.

The default state is no password protection. There is one password for each privilege level in the PLC. Each password may be unique; however, the same password can be used for more than one level. Passwords are one to seven ASCII characters in length. Only the programmer may change passwords.

PLC password protection can be used to restrict access to selected PLC functions. After passwords have been set up, access to the PLC via any communications path is restricted unless the proper password has been entered. Once a password has successfully been accepted, access to the privilege level requested and below will be granted (e.g., provide password for level 3 will allow access to functions at levels 0, 1, 2, and 3). If the PLC communications are suspended, protection level will automatically return to the lowest privilege level of 1) the highest unprotected level or 2) privilege level 2.

Priv Level	Password	Access Description
4	Yes	Write to all configuration or logic. Configuration may only be written in STOP mode; logic may be written in STOP or RUN mode. Set or delete passwords for any level.
3	Yes	Write to all configuration or logic when the CPU is in STOP mode, including word-for-word changes, the addition/deletion of program logic, and the overriding of discrete I/O.
2	Yes	Write to any data memory, this includes toggle/force of reference values but does **not** include overriding discrete I/O. The PLC can be started or stopped. PLC and I/O fault tables can be cleared. **NOTE:** This is the default if no passwords are defined.
1	Yes	Read any PLC data, except for passwords. This includes read fault tables, perform datagrams, verify logic/config, and load program and configuration from the PLC. NO PLC memory may be changed.
0	No	Read the current status of the PLC (including features supported by the PLC), read name of the Resource (CP name prior to release 6.0), change privilege level, and login as programmer.

Protection Level Request from Programmer

Upon connection to the CPU, the Programmer requests the protection status of each privilege level from the CPU. The Programmer then requests the CPU to move to the highest non-protected level, thereby giving the programmer access to the highest non-protected level without it having to specifically request any particular level.

A programmer requests a privilege level change by supplying the new privilege level and the password for that level. If the password sent by the programmer does not agree with the password stored in the PLCs password access table for the requested level, the privilege level change is denied and a fault is logged in the PLC fault table. The current privilege level is maintained, and no change will occur. A request to change to a privilege level that is not password protected is made by supplying the new level and a null (Hex 0) password. A privilege change may be to a lower level as well as to a higher level. Refer to chapter 5, "PLC Control and Status," in the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263, for instructions on displaying and changing passwords.

Disabling Passwords

The use of password protection is optional. If the user desires to prevent the use of password protection, passwords can be disabled using the Logicmaster 90-70 software. For instructions on how to enable or disable passwards, refer to chapter 5, "PLC Control and Status," in the *Logicmaster™ 90-70 Programming Software User's Manual,* GFK-0263.

Note

To reenable passwords after passwords have been disabled, the PLC must be power-cycled with the battery removed.

Password protection also prevents firmware upgrades to the FLASH memories used on the CPM 914, 915, 924, and 925. Prior to attempting a firmware upgrade in any of these modules, please disable password protection, then reenable it after the upgrade.

OEM Protection

OEM protection is similar to the passwords and privilege levels; however, OEM protection provides a higher level of security. The OEM protection feature is enabled/disabled using a 1 to 7 character password. When OEM protection is enabled, all read and write access to the PLC program and configuration is prohibited. Refer to chapter 5, "PLC Control and Status," in the *Logicmaster™ 90-70 Programming Software User's Manual,* GFK-0263, for instructions on enabling or disabling OEM protection.

Protection for OEMs' investment in software is provided in the form of a special password known as the *OEM key.* When the OEM key has been given a non-NULL value, the CPU may be placed in a mode in which reads and writes of the logic as well as writes to the configuration are prohibited. This allows a third-party OEM to create Control Programs for the PLC CPU and then set the OEM-locked mode which prevents the end-user from reading or modifying the program

Note

OEM protection also prevents firmware upgrades to the FLASH memories used on the CPM 914, 915, 924, and 925. Prior to attempting a firmware upgrade in any of these modules, please disable OEM protection, then enable it again after the upgrade.

Write Protect Keyswitch

The 90-70 CPU models 781,782, 788, 789, and 914 and higher CPUs all contain a memory write protect keyswitch. This keyswitch is located on the top of the faceplate, above the upper faceplate-to-rack clip. When in the protected position, the PLC program and configuration cannot be modified or deleted.

Note

The write protect keyswitch, when in the "write protected" position, also prevents firmware upgrades to the FLASH memories used on the CPM 914, 915, 924, and 925. Prior to attempting a firmware upgrade in any of these modules, please place the write protect keyswitch into the "write enabled" position.

Section 9: Series 90-70 PLC I/O System

The Series 90-70 PLC I/O system provides the interface between the Series 90-70 PLC and user-supplied devices and equipment. The I/O system supports the rack-type Model 70 I/O, the Genius I/O system, and the FIP I/O system. A Genius I/O Bus Controller (GBC) module provides the interface between the Series 90-70 PLC CPU and a Genius I/O bus. A FIP I/O Bus Controller (FBC) module provides the interface between the Series 90-70 PLC CPU and a FIP I/O bus. In addition to supporting these three I/O subsystems, the I/O system will also support Ethernet Interfaces and PCMs.

The I/O structure for the Series 90-70 PLC is shown in the following figure:

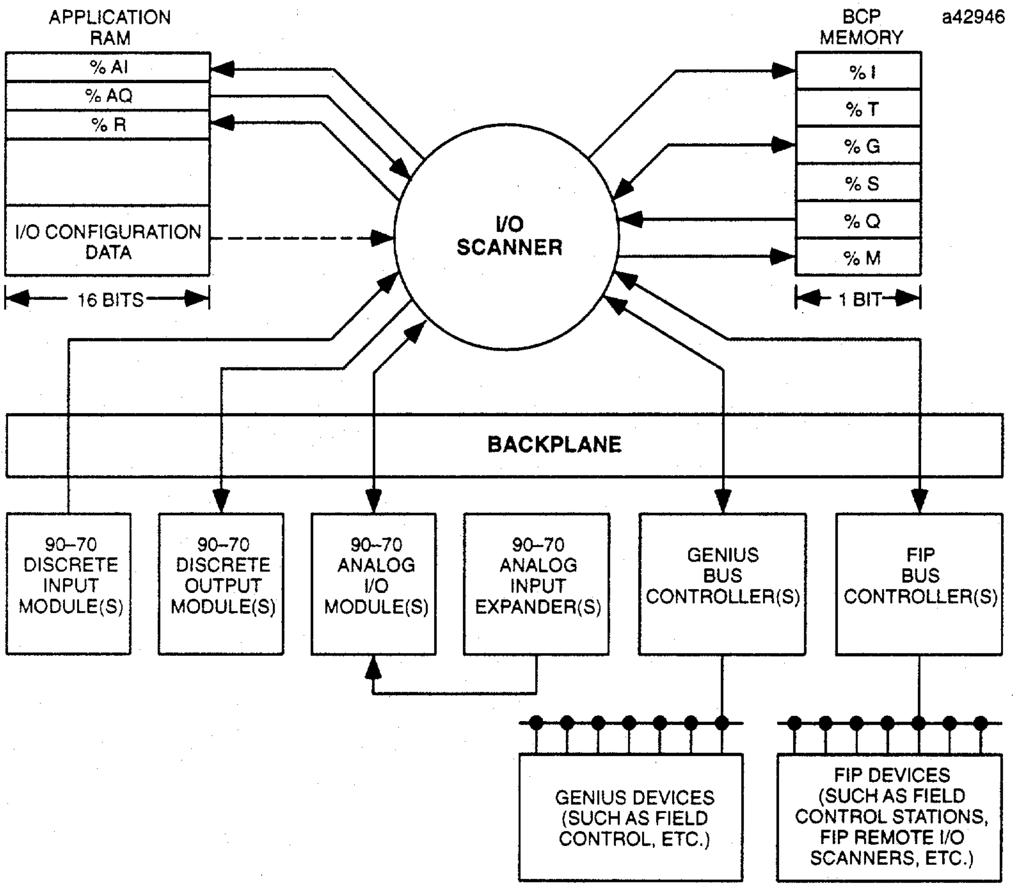

APPLICATION RAM
% AI
% AQ
% R
I/O CONFIGURATION DATA
16 BITS
BCP MEMORY
a42946
% I
% T
% G
% S
% Q
% M
1 BIT
I/O SCANNER
BACKPLANE
90–70 DISCRETE INPUT MODULE(S)
90–70 DISCRETE OUTPUT MODULE(S)
90–70 ANALOG I/O MODULE(S)
90–70 ANALOG INPUT EXPANDER(S)
GENIUS BUS CONTROLLER(S)
FIP BUS CONTROLLER(S)
GENIUS DEVICES (SUCH AS FIELD CONTROL, ETC.)
FIP DEVICES (SUCH AS FIELD CONTROL STATIONS, FIP REMOTE I/O SCANNERS, ETC.)

Figure 2-16. Series 90-70 PLC I/O Structure

Series 90-70 I/O Option Modules

Table 2-12 below lists the Series 90-70 I/O modules, including their catalog number, number of I/O points, module description, and the applicable data sheet or manual number for each module. For more information about a module, refer to the appropriate data sheet.

Table 2-15. Series 90-70 Option Module Types *

Catalog Number	I/O Points	Description	Data Sheet or Manual
		Discrete Input	
IC697MDL240	16	Input 120 VAC Isolated	GFK-0375
IC697MDL241	16	Input 240 VAC Isolated	GFK-0376
IC697MDL250	32	Input 120 VAC	GFK-0084
IC697MDL251	16	Input 120 VAC	GFK-0718
IC697MDL252	32	Input 12 VAC	GFK-0756
IC697MDL253	32	Input 24 VAC	GFK-0757
IC697MDL254	32	Input 48 VAC	GFK-0784
IC697MDL640	32	Input 125 VDC, Positive/Negative Logic	GFK-0719
IC697MDL650	32	Input 24 VDC, Positive Logic	GFK-0080
IC697MDL651	32	Input TTL	GFK-0377
IC697MDL652	32	Input 12 VDC, Positive/Negative Logic	GFK-0378
IC697MDL653	32	Input 24 VDC, Positive/Negative Logic	GFK-0379
IC697MDL654	32	Input 48 VDC, Positive/Negative Logic	GFK-0380
IC697MDL671	14	Interrupt Input, 24 VDC, Positive/Negative Logic	GFK-0880
		Discrete Output	
IC697MDL340	16	Output 120 VAC, 2A	GFK-0082
IC697MDL341	12	Output 120/240 VAC, 2A, Isolated	GFK-0382
IC697MDL350	32	Output 120 VAC, 0.5A	GFK-0081
IC697MDL740	16	Output 24/48 VDC, 2A, Positive Logic	GFK-0086
IC697MDL750	32	Output 24/48 VDC, 0.5A, Positive Logic	GFK-0085
IC697MDL752	32	Output 12 VDC 0.5A,	GFK-0381
IC697MDL753	32	Output 5 to 48 VDC, 0.5A	GFK-0383
IC697MDL940	16	Output Relay, 2A	GFK-0384
		Analog Modules	
IC697ALG230	–	Analog Input, High Level (8 Channels)	GFK-0385
IC697ALG440	–	Analog Expander, Current (16 Channels)	GFK-0385
IC697ALG441	–	Analog Expander, Voltage (16 Channels)	GFK-0385
IC697ALG320	–	Analog Output, Current/Voltage (4 Channels)	GFK-0388
		Intelligent Modules	
IC697ADC701	–	Alphanumeric Display Coprocessor	GFK-0521
IC697BEM731	–	Genius Bus Controller	GFK-0165
IC697CMM711	–	Communications Coprocessor Module	GFK-0370
IC697CMM741	–	Ethernet Interface	GFK-0532
IC697GDC701	–	Graphics Display Coprocessor	GFK-0519
IC697PCM711	–	Programmable Coprocessor Module	GFK-0164
IC697BEM733	–	Remote I/O Scanner	GFK-0539
IC697BEM721	–	I/O Link Interface	GFK-0645
IC697HSC700	–	High Speed Counter	GFK-1057
IC697BEM741	–	FIP Bus Controller	GFK-1002
IC670FBI001	–	FIP Bus Interface Unit	GFK-1175

* Some of the I/O modules listed above may not be available at the time this manual is printed. For current availability, consult your GE Fanuc PLC distributor or local GE Fanuc sales representative.

I/O Data Mapping

Discrete inputs and outputs are stored as bits in the CPU BCP Bit Cache memory. Analog I/O is stored in the application RAM allocated for that purpose. Analog data is always stored in the demultiplexed state, with each channel requiring one word (16 bits).

Default Conditions

The configuration utility provides the ability to specify that the first input may be an interrupt input and for the filter speed to be fast or slow; but upon power-up, Model 70 discrete input modules always default to the first input on the module not interrupting and the input filter being slow speed. If changed by the user, new defaults are applied when the board is configured by the CPU during the power-up process or whenever else the module may go through configuration.

Model 70 discrete output modules default to all outputs off. The configuration utility provides the ability to specify whether the CPU transitions from **`RUN/ENABLED`** to **`RUN/DISABLED`** or **`STOP`** mode. It also applies this default information when the system halts.

Genius I/O

Information relative to using Genius I/O in a Series 90-70 PLC system is presented in the following paragraphs. For specific information on Genius I/O block types, configuration, and setup, refer to the *Genius I/O System User's Manuals*, GEK-90486-1 and -2.

Genius I/O Bus Configuration

The Bus Controller used in the Series 90-70 PLC controls a single Genius I/O bus. Any type of Genius I/O block may be attached to the bus.

In the I/O fault table, the rack, slot, bus, module, and I/O point number are given for a fault. Bus number one refers to the bus on the single-channel Genius Bus Controller.

Genius I/O Data Mapping

Genius I/O discrete inputs and outputs are stored as bits in the CPU Bit Cache memory. Genius I/O analog data is stored in the application RAM allocated for that purpose (%AI and %AQ). Analog data is always stored one channel per one word (16 bit).

An analog grouped module consumes (in the input and output data memories) only the amount of data space required for the actual inputs and outputs. For example, the Genius I/O 115 VAC Grouped Analog Block, IC660CBA100, has four inputs and two outputs; it consumes four words of Analog Input memory (%AI) and two words of Analog Output memory.

A discrete grouped module, each point of which is configurable with the Hand-Held Monitor (HHM) to be input, output, or output with feedback, consumes an amount in both discrete input memory (%I) and discrete output memory (%Q) equal to its physical size. Therefore, the 8 I/O 115 VAC Discrete Grouped Block (IC660CBD100) requires 8 bits in the %I memory and 8 bits in the %Q memory, regardless of how the block is configured.

The following four Genius I/O blocks are assigned to the analog memories:

- 6-Channel Analog Grouped Block
- 6-Channel Thermocouple Block
- 6-Channel RTD Block
- 4-Channel Strain Gauge/mV Analog Input Block

The Thermocouple, RTD and Strain Gauge blocks are also referred to as Low-Level Analog Input blocks.

Analog Grouped Block

The Analog Grouped block contains four analog input channels and two analog output channels. When a block gets its turn on the Genius I/O Bus, it broadcasts the data for all four input channels in one broadcast control message. Then, when the Bus Controller gets its turn, it sends the data for both output channels to the block in a directed control message.

Low-Level Analog Blocks

Unlike the Analog Grouped block, the low-level analog blocks are input-only blocks. All have six channels except the Strain-Gauge block, which has four.

Default Conditions

Genius I/O blocks have a number of default conditions that may be set using the Genius I/O Hand-Held Monitor. These defaults include the following:

- Report faults
- Range select
- Analog input and output scaling
- Input filter time
- Alarm input mode
- Output hold last state
- Output default

These defaults are stored in EEPROM in the block itself. The Series 90-70 PLC configuration utility supports the changing of only a small subset of these defaults. For more information, refer to the *Genius I/O System User's Manuals*, GEK-90486-1 and -2.

Through the COMMREQ function block, the application program can request the Bus Controller to change any default condition on a specific block. However, this change will only be accepted by the block if it is not in **CONFIG PROTECT** mode. If **CONFIG PROTECT** mode is set, only the Hand-Held Monitor can be used to change the defaults. The format of the COMMREQ function block for Genius I/O is described in the *Genius Bus Controller User's Manual,* GFK-0398.

Genius Global Data Communications

The Series 90-70 PLC supports the sharing of data between multiple PLC systems that share a common Genius I/O bus. This mechanism provides a means for the automatic and repeated transfer of %G, %I, %Q, %AI, %AQ, and %R data. No special application programming is required to use global data since it is integrated into the I/O scan. All GE Fanuc PLCs that have Genius I/O capability can send global data to a Series 90-70 PLC and can receive data from a Series 90-70 PLC. Logicmaster 90 configuration software is used to configure the receiving and transmitting of global data on a Genius I/O bus.

Note

Genius global data communications do not continue to operate when the 90-70 PLC is in STOP/NOIO mode. However, if the 90-70 PLC is in STOP/IOSCAN mode, then Genius global data communications will continue to operate.

FIP I/O

Information relative to using FIP I/O in a Series 90-70 PLC system is presented in the following paragraphs. For specific information on FIP I/O types, configuration, and setup, refer to the *Series 90-70 FIP Bus Controller User's Manual,* GFK-1038.

FIP I/O Bus Configuration

The FIP Bus Controller used in the Series 90-70 PLC controls a single FIP I/O bus. Currently supported are the 90-30 FIP Remote I/O Scanner, FIP Bus Interface Unit (for Field Control), and generic FIP I/O module configurations. All of the FIP I/O interface modules (e.g., the FIP Remote I/O Scanner) must provide input data to the 90-70 FIP Bus Controller so that the FIP Bus Controller has this same input data to provide to the 90-70 CPU during the next normal input scan. Similarly, when the 90-70 CPU performs the next output scan, the FIP Bus Controller accepts this output data and passes it on to the appropriate FIP I/O interface module to then update the local I/O.

90-30 FIP Remote I/O Scanner

The FIP Remote I/O Scanner provides the ability to use 90-30 I/O as a remote I/O node on a FIP I/O network. The FIP Remote I/O Scanner module provides the communications interface to the FIP I/O network (communications with the 90-70 FIP Bus Controller) and also provides the I/O scanning function for the local 90-30 I/O modules. (For more information on the 90-30 FIP Remote I/O Scanner, please refer to the 90-30 FIP Remote I/O Scanner User Manual, GFK-1037)

FIP Bus Interface Unit (Field Control)

The FIP Bus Interface Unit provides the ability to use Field Control I/O as a remote I/O node on a FIP I/O network. The FIP Bus Interface Unit module provides the communication interface to the FIP I/O network (communications with the 90-70 FIP Bus Controller) and also provides the I/O scanning function for the local Field Control modules. (For more information on FIP Field Control, please refer to the *FIP Bus Interface Unit User's Manual,* GFK-1175).

Generic FIP I/O

Generic FIP I/O allows for configuring FIP I/O other than the FIP Remote I/O Scanner and FIP Bus Interface Unit. This permits the 90-70 CPU and the 90-70 FIP Bus Controller to assign I/O reference addresses to the generic FIP I/O device. The configuration selection also permits the 90-70 FIP Bus Controller to recognize the generic FIP I/O module on the FIP I/O network.

FIP I/O Fault Data

In the I/O fault table, the rack, slot, FIP drop id, remote rack, and remote slot number are given for faults occurring in an FIP Remote I/O Scanner or FIP Bus Interface Unit controlled remote I/O node. No fault information can be obtained from generic FIP I/O.

FIP I/O Data Mapping

FIP I/O discrete inputs and outputs are stored as bits in the CPU Bit Cache memory. FIP I/O analog data is stored in the application RAM allocated for that purpose (%AI and %AQ). Analog data is always stored one channel per one word (16 bit).

An analog grouped module consumes (in the input and output data memories) only the amount of data space required for the actual inputs and outputs. For example, an analog module with four inputs and two outputs consumes four words of Analog Input memory (%AI) and two words of Analog Output memory.

Default Conditions

FIP I/O devices have a number of default conditions which may be set using the Series 90-70 Logicmaster Configuration Utility. The default conditions include the following:

- Range select
- Analog input scaling
- Analog output scaling
- Discrete output default OFF (fixed)
- Analog output default HOLD LAST STATE (fixed)

Diagnostic Data Collection

Diagnostic data in a Series 90-70 PLC I/O system is obtained in one of the following two ways:

1. If an I/O module has an associated Bus Controller (Genius Bus Controller or FIP Bus Controller), then the Bus Controller provides the module's diagnostic data for the CPU.
2. If an I/O module is a Model 70 I/O module, then the CPU's I/O Scanner subsystem generates the diagnostic bits based on the data provided by the I/O module.

The diagnostic bits are derived from the diagnostic data sent from the I/O modules to their I/O controllers (Genius Bus Controller, FIP Bus Controller, or 90-70 CPU).

Diagnostic bits always indicate the current fault status of the associated module. Bits are set when faults occur and are cleared when faults are cleared.

In general, diagnostic data is not maintained by the Series 90-70 PLC for foreign I/O (not GE Fanuc) modules. Any diagnostic information provided by those boards must be specifically accessed by the application program using the VME Read and VME Write function blocks.

Beginning with 90-70 CPU release 5.50, the 90-70 system has supported foreign I/O modules when developed under license agreement with GE Fanuc. These boards are then configured as "FOREIGN VME" and the interface mode is "I/O SCAN". Boards developed to conform to the I/O Scan interface can provide discrete and analog diagnostic information to the 90-70 CPU.

Discrete I/O Diagnostic Information

Diagnostic information is maintained by the Series 90-70 PLC for each discrete I/O point. Two memory blocks are allocated in application RAM for discrete diagnostic data. One is associated with %I memory and the other with %Q memory. One bit of diagnostic memory is associated with each I/O point. This bit indicates the validity of the associated I/O data. Each discrete point has a fault reference available that may be interrogated using two special contacts: a fault contact (-[FAULT]-) and a no-fault contact (-[NOFLT]-). The PLC only collects this fault data if enabled to do so through the configuration software. The following table shows the state of the fault and no-fault contacts.

Condition	[FAULT]	[NOFLT]
Fault Present	ON	OFF
Fault Absent	OFF	ON

Analog I/O Diagnostic Data

Diagnostic information is made available by the PLC CPU for each analog channel associated with Model 70 analog input modules, Model 70 analog output modules, Genius analog blocks, etc. Two memory blocks are allocated for analog diagnostic data. One is associated with %AI analog input memory and the other with %AQ analog output memory. One byte of diagnostic memory is allocated for each analog I/O channel. Since each analog I/O channel uses two bytes of %AI and %AQ memory, the diagnostic memory is half the size of the data memory.

The analog diagnostic data contains both diagnostics and process data with the process data being the High Alarm and Low Alarm bits. The diagnostic data is referenced with the -[FAULT]- and -[NOFLT]- contacts. The process bits are referenced with the -[HIALR]- and -[LOALR]- contacts. The memory allocation for analog diagnostic data is one byte per word of analog input and analog output allocated by the user. When an analog fault contact is referenced in the application program, the PLC does an Inclusive OR on all the bits in the diagnostic byte except the process bits. The alarm contact is closed if any diagnostic bit is ON and OFF, only if all bits are OFF.

Chapter 3 Fault Explanation and Correction

This chapter is an aid to troubleshooting a Series 90-70 PLC system using Logicmaster 90-70 software. It explains the fault descriptions, which appear in the PLC fault table, and the fault categories, which appear in the I/O fault table.

Each fault explanation in this chapter lists the fault description for the PLC fault table or the fault category for the I/O fault table. Find the fault description or fault category corresponding to the entry on the applicable fault table displayed on your programmer screen. Beneath it is a description of the cause of the fault along with instructions to correct the fault.

Chapter 3 contains the following sections:

Section	Title	Description	Page
1	System Handling of Faults (General)	Provides a description of the PLC system faults (SY_FLT) and the I/O faults (IO_FLT). Provides a description of configurable fault and changing the fault action, non-configurable faults, and locating fault references (rack, slot, bus, and FIP locating references.	3-2
2	Fault Handling	Describes the type of faults that may occur in the Series 90-70 PLC and how they are displayed in the fault tables. Descriptions of the PLC and I/O fault table displays are also included, as well as how to access additional fault information by pressing **CTRL-F**.	3-11
3	PLC Fault Table Explanations	Provides a fault description of each PLC fault and instructions to correct the fault.	3-18
4	I/O Fault Table Explanations	Provides a description of each I/O fault and instructions to correct the fault.	3-39

Section 1: System Handling of Faults (General)

The system fault references listed below can be used to identify the specific type of fault that has occurred.

System Fault Reference	Description
ANY_FLT	Any fault in the system.
SY_FLT	Any system fault in the Series 90 PLC.
IO_FLT	Any I/O fault.
SY_PRES	Indicates a new entry in the PLC fault table.
IO_PRES	Indicates a new entry in the I/O fault table.
HRD_FLT	Any hardware fault.
SFT_FLT	Any software fault.

On power-up, the system fault references are cleared. If a fault occurs, the on-transition state of the affected reference(s) is on the next sweep after the fault occurs. The system fault references remain on as long as the fault exists, until the PLC is cleared or until cleared from the program. The ANY_FLT fault is set when any other fault is set. The SY_PRES and IO_PRES faults are set when the PLC and I/O fault table contain entries.

System Fault References

Additional information on the fault tables may be found in chapter 5, "PLC Control and Status," in the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263.

When a system fault reference is set, additional fault references are also set. The following table lists these other types of faults. References marked with asterisks below are configurable system fault references.

	PLC System Fault (SY_FLT)	I/O Fault (IO_FLT)
Hardware Fault (HRD_FLT)	SBUS_ER System bus error.	
	HRD_CPU PLC CPU hardware fault.	
	HRD_SIO Module hardware fault.	
	SBUS_FL System bus failure. *	
Software Fault (SFT_FLT)	SFT_SIO Option module software flt.	SFT_IOC Bus Controller software fault.
	SFT_CPU PLC software fault. *	
	MAX_IOC Too many Bus Controllers. *	
	STOR_ER Programmer download failed. *	
Other Faults	PB_SUM block checksum fault.	LOS_IOC Loss of Bus Controller.
	LOW_BAT Low battery signal.	LOS_IOM Loss of I/O module.
	OV_SWP Over constant sweep time.	ADD_IOC Addition of bus controller.
	SY_FULL PLC fault table full.	ADD_IOM Addition of I/O module.
	IO_FULL I/O fault table full.	IOC_FLT Bus or Bus Controller fault.
	APL_FLT Application program fault.	IOM_FLT I/O module fault.
	NO_PROG No application program at power-up. *	
	BAD_RAM Corrupted program memory. *	
	WIND_ER Incomplete window service. *	
	BAD_PWD Password access failure. *	
	NUL_CFG No configuration present. *	
	LOS_SIO Loss of option module. *	
	ADD_RCK Addition of expansion rack. *	
	ADD_SIO Addition of option module. *	
	CFG_MM Configuration mismatch. *	
	LOS_RCK Loss of rack. *	

* Configurable system fault references.

Configurable Fault Actions

For some faults, the PLC must stop execution. For other faults, the appropriate response to a fault may depend on the nature of the application. All faults are initially assigned to one of these three actions:

Fault Action	Description
Fatal	These faults halt the system, set diagnostic variables, and are logged in a fault table.
Diagnostic	These faults do not halt the system. They do, however, set diagnostic variables and are logged in a fault table.
Informational	These faults are logged in a fault table, but cause no other action.

For some faults, called "non-configurable" faults, the fault action cannot be changed. Other faults, called "configurable" faults, can have their fault type changed to another fault action if such a change is suitable for the application.

The following table lists configurable faults. If the fault also causes additional faults, only the initiating fault is logged.

Table 3-1. Configurable Fault References

Fault (Default Action)	Description	May Also Be Set
SBUS_ER (diagnostic)	System bus error. (The BSERR* signal was generated on the VME system bus.)	HRD_FLT SY_PRES, SY_FLT Other references may also be set depending on the type of access when the BSERR* occurred.
HRD_CPU (fatal)	PLC CPU hardware fault, such as failed memory device or failed serial port).	SY_FLT, SY_PRES HRD_FLT
HRD_SIO (diagnostic)	Non-fatal hardware fault on any module in the system, such as failure of a serial port on a PCM.	SY_FLT, SY_PRES HRD_FLT
SFT_IOC (diagnostic)	Non-recoverable software error in a Genius Bus Controller.	IO_FLT, IO_PRES SFT_FLT
SFT_SIO (diagnostic)	Non-recoverable software error in a PCM or LAN interface module.	SY_FLT, SY_PRES SFT_FLT
PB_SUM (fatal)	Program or block checksum failure during power-up or in **RUN** mode.	SY_FLT, SY_PRES
LOW_BAT (diagnostic)	Low battery signal from CPU or another module in system.	SY_FLT, SY_PRES
OV_SWP (diagnostic)	Constant sweep time exceeded.	SY_FLT, SY_PRES
SY_FULL IO_FULL (diagnostic)	PLC fault table full (16 entries). I/O fault table full (32 entries).	SY_FLT, SY_PRES IO_FLT, IO_PRES

Table 3-1. Configurable Fault References (cont'd)

Fault (Default Action)	Description	May Also Be Set
APL_FLT (diagnostic)	Application fault.	SY_FLT, SY_PRES
LOS_RCK (diagnostic)	Loss of rack (BRM failure, loss of power), or missing a configured rack.	SY_FLT, SY_PRES IO_FLT, IO_PRES
LOS_IOC (diagnostic)	Loss of Bus Controller channel, or missing a configured Bus Controller.	IO_FLT, IO_PRES
LOS_IOM (diagnostic)	Loss of I/O module (does not respond), or missing a configured I/O module.	IO_FLT, IO_PRES
LOS_SIO (diagnostic)	Loss of option module (does not respond), or missing a configured module.	SY_FLT, SY_PRES
ADD_RCK (diagnostic)	New rack added, or previously faulted rack has returned.	SY_FLT, SY_PRES
ADD_IOC (diagnostic)	Previously faulted Bus Controller is no longer faulted.	IO_FLT, IO_PRES
ADD_IOM (diagnostic)	Previously faulted I/O module is no longer faulted.	IO_FLT, IO_PRES
ADD_SIO (diagnostic)	New option module is added, or previously faulted module no longer faulted.	SY_FLT, SY_PRES
IOC_FLT (diagnostic)	Non-fatal bus or Bus Controller error, more than 10 bus errors in 10 seconds (error rate is configurable).	IO_FLT, IO_PRES
IOM_FLT (diagnostic)	Point or channel on an I/O module; a partial failure of the module.	IO_FLT, IO_PRES
CFG_MM (fatal)	Wrong module type detected during power-up or **RUN** mode. The PLC does not check the configuration parameters set up for individual modules such as Genius I/O blocks.	SY_FLT, SY_PRES

Non-Configurable Faults

For non-configurable faults, the fault action cannot be changed. These faults must always be either fatal or informational faults.

The following table lists non-configurable faults. If the fault also causes additional faults, only the initiating fault is logged.

Table 3-2. Non-Configurable Faults

Fault	Description	Result
SBUS_FL (fatal)	System bus failure. The PLC CPU was not able to access the VME bus. BUSGRT*NMI error.	Sets SY_FLT, HRD_FLT, and SY_PRES.
NO_PROG (information)	No application program is present at power-up. Should only occur the first time the PLC is powered up or if the battery-backed RAM containing the program fails.	Does not set any references. PLC will not go to **RUN** mode; it continues executing **STOP** mode sweep until a valid program is loaded. This can be a "null" program that does nothing. Sets SY_FLT and SY_PRES.
BAD_RAM (fatal)	Corrupted program memory at power-up. Program could not be read and/or did not pass checksum tests.	Sets SY_FLT and SY_PRES.
WIND_ER (information)	Window completion error. Servicing of Programmer or Logic Window was skipped. Occurs in **CONSTANT SWEEP** or **MICROCYCLE SWEEP** mode.	Sets SY_FLT and SY_PRES.
BAD_PWD (information)	Change of privilege level request to a protection level was denied; bad password.	Sets SY_FLT and SY_PRES.
NUL_CFG (fatal)	No configuration present upon transition to **RUN** mode. Running without a configuration is equivalent to suspending the I/O scans.	Sets SY_FLT and SY_PRES.
SFT_CPU (fatal)	CPU software fault. A non-recoverable error has been detected in the CPU. May be caused by Watchdog Timer expiring.	PLC immediately transitions to **ERROR SWEEP** mode. The only activity permitted is communication with the programmer. To be cleared, PLC power must be cycled. Sets SY_FLT, SY_PRES, and SFT_FLT.
MAX_IOC (fatal)	The maximum number of bus controllers has been exceeded. The Series 90 PLC supports 32 bus controllers.	Sets SY_FLT, SY_PRES, and SFT_FLT.
STOR_ER (fatal)	Download of data to PLC from the programmer failed; some data in PLC may be corrupted.	PLC will not transition to **RUN** mode. This fault is not cleared at power-up, intervention is required to correct it. Sets SY_FLT and SY_PRES.

Fault Locating References (Rack, Slot, Bus)

The Series 90-70 PLC supports reserved fault names for each rack, slot, bus, and module if the hardware is configured. By programming these names on the FAULT and NOFLT contact instructions, logic can be executed to locate faults associated with configured racks and modules.

These fault nicknames can only be programmed on the FAULT and NOFLT contacts. The reserved fault names are always available. It is not necessary to enable a special option, such as point faults.

These fault names do not correspond to %SA, %SB, %SC, or to any other reference type. Only the nickname is displayed with Logicmaster 90 software. A reference table screen is not provided for the fault references.

The format of a rack fault nickname is RACK_0r, where r is the rack number 0 to 7. For example, RACK_01 shown in the example below represents rack 1.

The format of a slot fault nickname is SLOT_rs, where r is the rack number 0 to 7 and s is the slot number 0 to 9. For example, SLOT_15 shown in the example below represents rack 1, slot 5.

```
|
| RACK_01 SLOT_15                                                       %Q00002
|-[FAULT]-[NOFLT]-----------------------------------------------------------( )-
|
```

The format of a bus fault nickname is a BUS_rsb, where r is the rack number 0 to 7, s is the slot number 0 to 9, and b is the bus number 1 or 2. For example, BUS_241 represents rack 2, slot 4, bus 1.

The format of a module fault nickname is M_rsbmm, where r is the rack number 0 to 7, s is the slot number 0 to 9, b is the bus number 1 or 2, and mm is the module number 00 to 31. For example, M_26128 represents rack 2, slot 6, bus 1, module 28.

At power-up, all faults are cleared in the PLC. When a fault is logged, the PLC transitions the state of the affected reference(s). The state of the fault reference remains in the **FAULT** state until one of the following actions occurs:

- Both the PLC and the I/O fault tables are cleared through Logicmaster either by clearing each table individually or clearing the entire PLC memory.
- The associated device (rack, I/O module, or Genius device) is added back into the system. Whenever an "**ADDITION OF** . . . " fault is logged, the PLC initializes all fault references associated with the device to the **NOFLT** state. These references remain in the **NOFLT** state until another fault associated with the device is reported. (This could take several seconds for distributed I/O faults, especially if the bus controller has been reset.)

These fault references are set for informational purposes only. They should not be used to qualify I/O data. The I/O point fault references (described on the next page) may be used to qualify I/O data. The PLC does not halt execution as a result of setting a fault locating reference to the **FAULT** state.

The fault references have a cascading effect. If there is a problem in the module located at rack 5, slot 6, bus 1, module 29, the following faults references are set: RACK_05, SLOT_56, BUS_561, and M_56129. There will only be one entry in the fault table to describe the problem with the module. The fault table does not show entries pertaining to the rack, slot, and bus in this case.

Fault Contacts

Fault (-[FAULT]-) and no-fault (-[NOFLT]-) contacts are used to detect faults in discrete or analog machine references. These contacts can be forced but not overridden. The following table shows the state of fault and no-fault contacts.

Condition	[FAULT]	[NOFLT]
Fault Present	ON	OFF
Fault Absent	OFF	ON

Alarm Contacts

High (-[HIALR]-) and low (-[LOALR]-) alarm contacts are used to represent the state-of-the-art analog input module comparator function. However, the use of point faults must first be enabled in the configuration software by selecting CPU Configuration (F2) from the Configuration main menu and then selecting Memory Allocation and Point Fault Enable (F4) from the CPU Configuration menu. Change the *Point Fault Reference* setting from **DISABLED** to **ENABLED**.

The following example logic uses both high and low alarm contacts.

```
|%AI0001                                                                   %Q00003
|-[HIALR]+-----------------------------------------------------------------( )-
|        |
|%AI0002 |
|-[LOALR]+
|
```

Note

HIALR and LOALR contacts will not create an entry in a fault table.

Point Faults

Point faults pertain to external I/O faults, although they will also be set due to the failure of associated higher-level internal hardware (e.g., IOC failure or loss of a rack). In order to use point faults, they must be enabled in the configuration software by selecting CPU Configuration (F2) from the Configuration main menu and then selecting Memory Allocation and Point Fault Enable (F4) from the CPU Configuration menu. Change the *Point Fault Reference* setting from **DISABLED** to **ENABLED**.

When enabled, a boolean reference for each discrete I/O point fault and a byte reference for each analog I/O channel level fault are allocated, and are traded off against the space available for the logic program. Whenever a point fault is generated, the system level I/O fault reference (IO_FLT) is set. The FAULT and NOFLT contacts described above provide access to the point fault.

I/O fault reference (IO_FLT) is set. The FAULT and NOFLT contacts described above provide access to the point fault.

FIP Device Fault Locating References

The Series 90-70 PLC supports fault reference nicknames for FIP devices. Refer to the table and note presented below for information when dealing with FIP fault locating references.

Fault Reference Type	Reserved Nickname	Comment
Rack	RACK_0r	where r is rack number 0 to 7.
Slot	SLOT_rs	where r is rack number 0 to 7 and s is slot number 0 to 9.
Bus	BUS_rsb	where r is rack number 0 to 7, s is slot number 0 to 9, and b is bus number 1 or 2.
Module	M_rsbmm	where r is rack number 0 to 7, s is slot number 0 to 9, b is bus number 1 or 2, and mm is the module number 00 to 31.
FIP Module	F_rsmmm	where r is rack number 0 to 7, s is slot number 0 to 9, and mmm is the FIP module number 00 to 255.

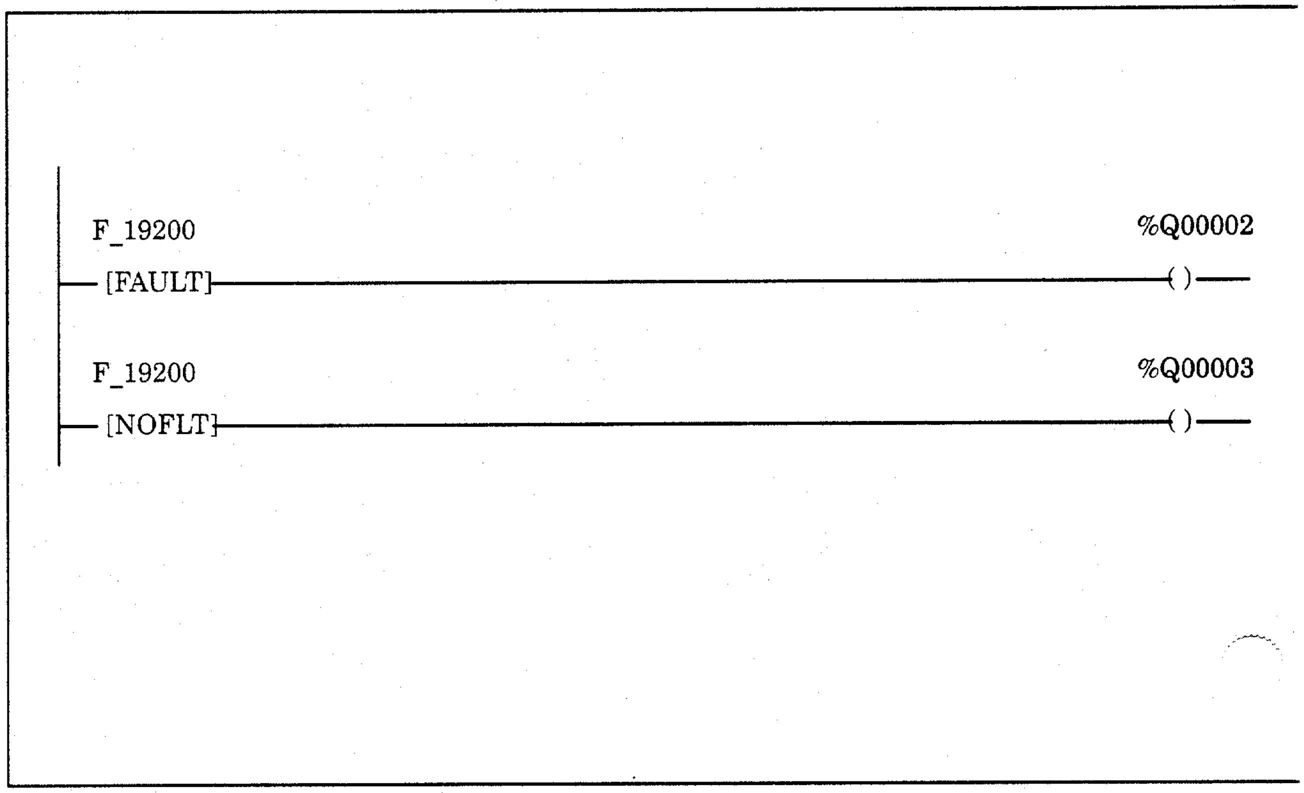
Note

Bus Type fault references will not set for FIP devices. Bus and Module Type Fault References are only set for Genius devices. F_rsmmm references will not be affected for non-FIP devices. FIP fault references have a cascading effect. A fault in FIP device at rack 5, slot 6, module 29 results in the following fault references being set: RACK_05, SLOT_56, F_56029.

A module fault for a FIP Remote I/O Scanner device results in only one Module Fault Type Fault Reference being set. For example, if FIP module with station ID 43 in 90-70 rack 3, slot 9 is an FIP Remote I/O Scanner, and within the FIP Remote I/O Scanner there is a faulted module in the FIP Remote I/O Scanner's rack 1, slot 5 , then only Fault Reference F_39043 will be set.

FAULT/NOFLT Contact Instructions

The FAULT and NOFLT contact instructions will accept the Nickname F_rsmmm (see table on the previous page). For example, to detect a fault for a FIP device number 200 in rack 1, slot 9, the following rungs could be entered:



The functionality of the FIP device module fault references is the same as the other module fault references. The state of the fault reference will remain as long as the fault exists. Clearing the PLC will reset all faults to their initial state. For the above example, the following additional fault references will be set: RACK_1 and SLOT_19.

Section 2: Fault Handling

Note

This information on fault handling applies to systems programmed using Logicmaster 90-70 software.

Faults occur in the Series 90-70 PLC system when certain failures or conditions happen which affect the operation and performance of the system. These conditions, such as the loss of an I/O module or rack, may affect the ability of the PLC to control a machine or process. These conditions may also have beneficial effects, such as when a new module comes online and is now available for use. Or, these conditions may only act as an alert, such as a low battery signal which indicates that the battery protecting the memory needs to be changed.

For information on system status/fault references, refer to chapter 2, section 3, "Program Organization and User Data."

Alarm Processor

The condition or failure itself is called a fault. When a fault is received and processed by the CPU, it is called an alarm. The software in the CPU which handles these conditions is the Alarm Processor. The interface to the user for the Alarm Processor is through Logicmaster 90-70 programming software. Any detected fault is recorded in a fault table and displayed on either the PLC fault table screen or the I/O fault table screen, as applicable.

Classes of Faults

The Series 90-70 PLC detects several classes of faults. These include internal failures, external failures, and operational failures.

Table 3-3. Classes of Faults

Fault Class	Examples
Internal Failures	Non-responding modules. Low battery condition. Memory checksum errors.
External I/O Failures	Loss of rack or module. Addition of rack or module. Loss of Genius I/O block.
Operational Failures	Communication failures. Configuration failures. Password access failures.

System Reaction to Faults

Typically, hardware failures require that either the system be shut down or the failure be tolerated. I/O failures may be tolerated by the PLC system, but they may be intolerable by the application or the process being controlled. Operational failures are normally tolerated. Series 90-70 PLC faults have three attributes:

Table 3-4. Fault Attributes

Attribute	Description
Fault Table Affected	I/O fault table PLC fault table
Fault Action	Fatal Diagnostic Informational
Fault Response	Configurable Non-configurable

Fault Tables

Two fault tables are provided to make faults easier to find and to keep a single table from becoming too long. These tables are the PLC fault table and the I/O fault table. An explanation of the PLC fault table begins on page 3-5. An explanation of the I/O fault table begins on page 3-6.

Fault Action

Fatal faults cause the fault to be recorded in the appropriate table, any diagnostic variables to be set, and the system to be stopped. Diagnostic faults are recorded in the appropriate table, and any diagnostic variables are set. Informational faults are only recorded in the appropriate table.

Table 3-5. Fault Actions

Fault Action	Response by CPU
Fatal	Log fault in fault table. Set fault references. Go to **STOP** mode.
Diagnostic	Log fault in fault table. Set fault references.
Informational	Log fault in fault table.

The configuration utility provides the capability to change the fault action of certain faults. There are two possible classifications in the utility: fatal and non-fatal. These correspond to fatal and diagnostic fault action in the PLC. Only fatal faults cause the system to halt. Additionally, the informational fault action only logs faults in the fault table.

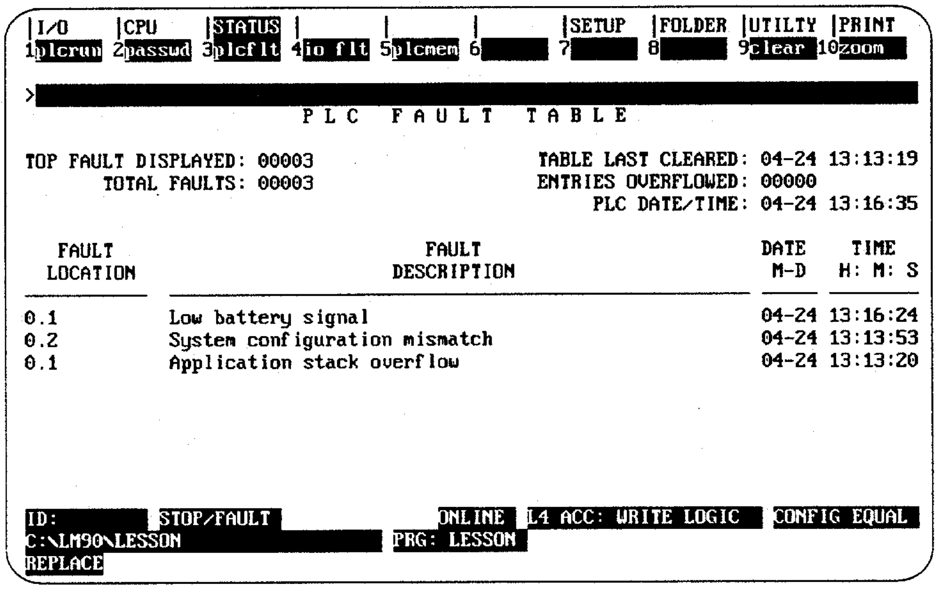
When a fault is detected by the CPU, it uses a default fault action for that fault. For maskable faults, the CPU uses the fault action specified by the configuration utility; this may be the default or a fault action chosen by the user.

Fault Response

Fault response refers to the ability of a fault to have its fault action changed. Those faults which can have their fault action changed are called configurable faults. Those which cannot are called non-configurable faults. Non-configurable faults are either fatal or informational. Also, non-configurable faults do not cause application available references to be set and cannot have alarm blocks associated with the detection of the fault. Some non-configurable faults also have other effects associated with them. Generally, these effects control the changing of the CPUs execution mode (**STOP**, **RUN/DISABLED**, **RUN/ENABLED**). An example of such an effect is the disabling of I/O when a null system configuration is detected in the system.

PLC Fault Table

The PLC fault table displays PLC faults such as password violations, PLC/configuration mismatches, parity errors, and communications errors. For example:



```
|I/O      |CPU     |STATUS  |        |        |        |SETUP   |FOLDER  |UTILITY |PRINT
1plcrun  2passwd  3plcflt  4io flt  5plcmem  6        7        8        9clear  10zoom

>
                          P L C    F A U L T    T A B L E

TOP FAULT DISPLAYED: 00003                        TABLE LAST CLEARED: 04-24 13:13:19
      TOTAL FAULTS: 00003                        ENTRIES OVERFLOWED: 00000
                                                     PLC DATE/TIME: 04-24 13:16:35

  FAULT                       FAULT                                  DATE    TIME
 LOCATION                  DESCRIPTION                               M-D   H: M: S

0.1          Low battery signal                                     04-24 13:16:24
0.2          System configuration mismatch                          04-24 13:13:53
0.1          Application stack overflow                             04-24 13:13:20



ID:          STOP/FAULT                 ONLINE  L4 ACC: WRITE LOGIC   CONFIG EQUAL
C:\LM90\LESSON                         PRG: LESSON
REPLACE
```

To display a screen similar to the one shown above, press **PLC Fault (F3)** from the PLC Control and Status menu or from another PLC functions screen. The programmer may be in any operating mode. However, if the programmer is in **OFFLINE** mode, no faults are displayed. In **ONLINE** or **MONITOR** mode, PLC fault data is displayed. In **ONLINE** mode, faults can be cleared (this may be password protected).

Field	Description
Top Fault Displayed	The index of the PLC fault currently at the top of the fault display is shown on the first line of this screen.
Total Faults	The total number of faults since the table was last cleared.
Table Last Cleared	The date and time faults were last cleared from the fault table. This information is maintained by the PLC.
Entries Overflowed	The number of entries lost because the fault table has overflowed since it was cleared. The PLC fault table can contain up to 16 faults.
PLC Time/Date	The current date and time. This is also maintained by the PLC.

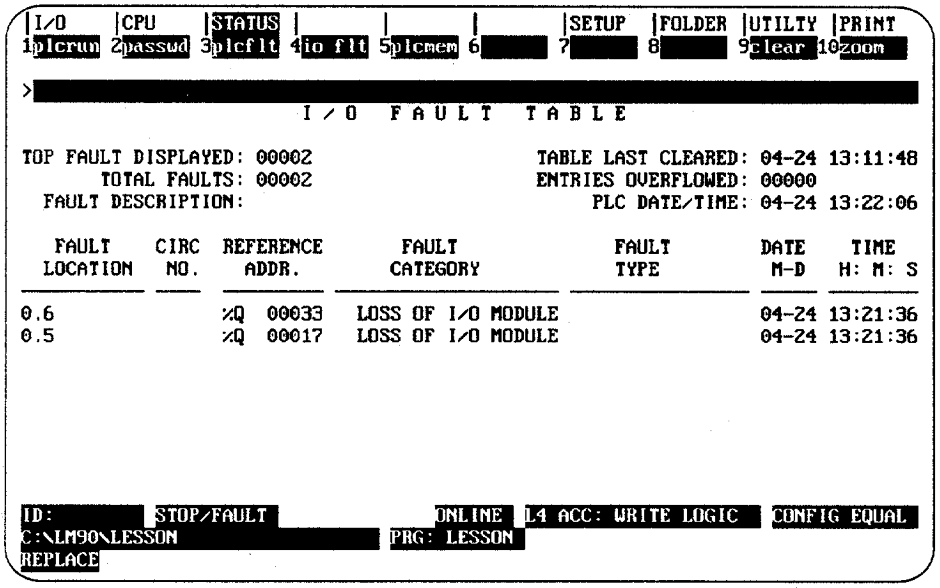
For more information on the PLC fault table, refer to the chapter 5, "PLC Control and Status," in the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263.

Note

The PLC fault table can contain up to 16 faults. Additional faults cause the table to overflow, and faults are lost. The system reference SY_FULL (%S0009) is set to indicate that the fault table is full. For detailed information about this, refer to the "Number of Faults in the PLC Fault Table" section of chapter 5, "PLC Control and Status," in the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263.

I/O Fault Table

The I/O fault table displays I/O faults such as circuit faults, address conflicts, forced circuits, and I/O bus faults. For example:



```
|I/O      |CPU     |STATUS |       |       |       |SETUP   |FOLDER  |UTILTY  |PRINT
1plcrun  2passwd  3plcflt 4io flt 5plcmem 6        7        8        9clear  10zoom

>
                          I / O    F A U L T    T A B L E

TOP FAULT DISPLAYED: 00002                         TABLE LAST CLEARED: 04-24 13:11:48
       TOTAL FAULTS: 00002                        ENTRIES OVERFLOWED: 00000
  FAULT DESCRIPTION:                                   PLC DATE/TIME: 04-24 13:22:06

   FAULT     CIRC  REFERENCE       FAULT                  FAULT          DATE    TIME
  LOCATION   NO.     ADDR.        CATEGORY                TYPE           M-D    H: M: S

0.6                %Q  00033  LOSS OF I/O MODULE                         04-24 13:21:36
0.5                %Q  00017  LOSS OF I/O MODULE                         04-24 13:21:36

ID:         STOP/FAULT                  ONLINE  L4 ACC: WRITE LOGIC    CONFIG EQUAL
C:\LM90\LESSON                       PRG: LESSON
REPLACE
```

To display a screen similar to the one shown above, press **I/O Fault (F4)** from the PLC Control and Status menu or from another PLC functions screen. The programmer may be in any operating mode. However, if the programmer is in **OFFLINE** mode, no faults are displayed. In **ONLINE** or **MONITOR** mode, PLC fault data is displayed. In **ONLINE** mode, faults can be cleared (this feature may be password protected).

Field	Description
Top Fault Displayed	The index of the I/O fault currently at the top of the fault display is shown on the first line of this screen.
Total Faults	The total number of faults since the table was last cleared.
Fault Description	An explanation of the fault that is currently highlighted in the I/O fault table.
Table Last Cleared	The date and time faults were last cleared from the fault table. This information is maintained by the PLC.
Entries Overflowed	The number of entries lost because the fault table has overflowed since it was cleared. The I/O fault table can contain up to 32 faults.
PLC Time/Date	The current date and time. This is also maintained by the PLC.

For more information on the I/O fault table, refer to the chapter 5, "PLC Control and Status," in the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263.

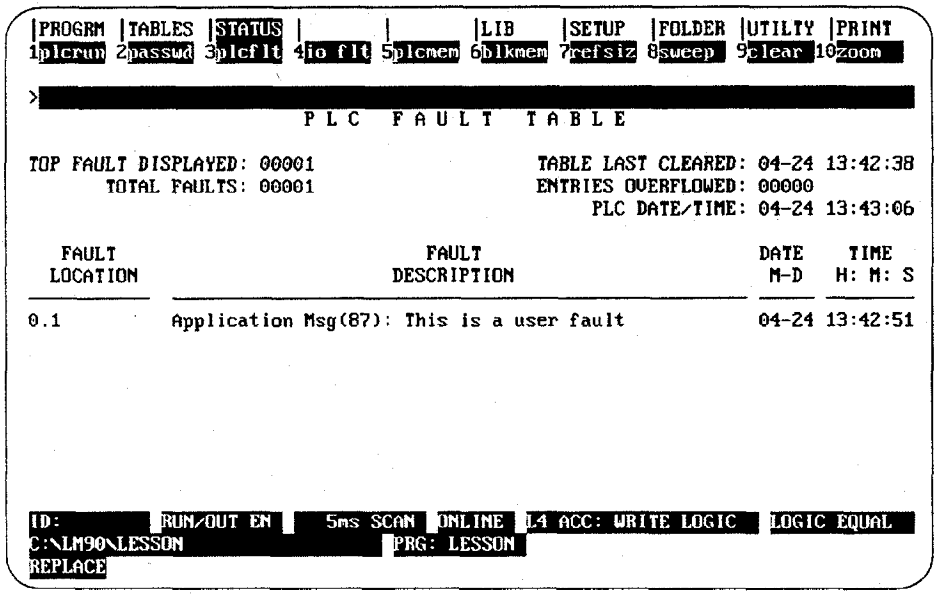
Note

The I/O fault table can contain up to 32 faults. Additional faults cause the table to overflow, and faults are lost. The system reference IO_FULL (%S0010) is set to indicate that the fault table is full. For detailed information about this, refer to the "Number of Faults in the I/O Fault Table" section of chapter 5, "PLC Control and Status," in the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263.

User-Defined Faults

User-defined faults can be logged in the PLC or I/O fault table. When a user-defined fault occurs, it is logged in the appropriate fault table as "Application Msg(error_code):" and may be followed by a descriptive message up to 24 characters. All characters in the descriptive message can be defined by the user. Although the message must end with the null character, e.g., zero (0), the null character does not count as one of the 24 characters. If the message contains more than 24 characters, only the first 24 characters are displayed.

The following example PLC fault table contains a user-defined fault with error code 87.


```
|PROGRM |TABLES |STATUS |       |       |LIB    |SETUP  |FOLDER |UTILTY |PRINT
1plcrun 2passwd 3plcflt 4io flt 5plcmem 6blkmem 7refsiz 8sweep  9clear 10zoom

>
                           P L C   F A U L T   T A B L E

TOP FAULT DISPLAYED: 00001                          TABLE LAST CLEARED: 04-24 13:42:38
       TOTAL FAULTS: 00001                          ENTRIES OVERFLOWED: 00000
                                                        PLC DATE/TIME: 04-24 13:43:06

  FAULT                              FAULT                              DATE    TIME
 LOCATION                         DESCRIPTION                            M-D   H: M: S

0.1          Application Msg(87): This is a user fault                  04-24  13:42:51




ID:           RUN/OUT EN     5ms SCAN  ONLINE  L4 ACC: WRITE LOGIC    LOGIC EQUAL
C:\LM90\LESSON                        PRG: LESSON
REPLACE
```

For more information on user-defined faults, refer to the chapter 5, "PLC Control and Status," in the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263.

Note

For details about user-defined fault logging, refer to the information about Service Request # 21 on page 4-226 and following in this manual.

Accessing Additional Fault Information

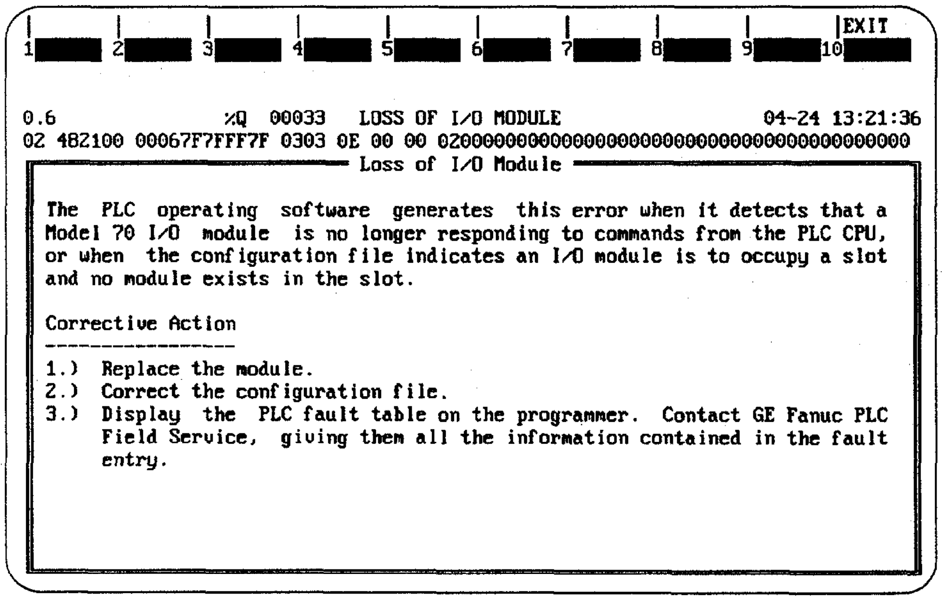
The fault tables displayed by Logicmaster 90-70 software contain basic information regarding the fault. Additional information pertaining to each fault can be displayed by positioning the cursor on the fault entry and pressing the **Zoom (F10)** softkey from the PLC or I/O fault table screen. For more information about this fault zoom feature, refer to chapter 5, "PLC Control and Status," in the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263. In addition, a hexadecimal dump of the fault can be obtained by positioning the cursor on the fault entry and pressing the **CTRL-F** key sequence. For more information about using CTRL-F, refer to appendix B, "Interpreting Fault Tables Using Logicmaster 90-70 Software," in this manual.

The last entry, ***Correction,*** for each fault explanation in this chapter lists the action(s) to be taken to correct the fault. Note that the corrective action for some of the faults includes the statement:

Display the PLC fault table on the programmer. Contact GE Fanuc PLC Field Service, giving them all the information contained in the fault entry.

This second statement means that you must tell Field Service both the information readable directly from the fault table <u>and</u> the hexadecimal information you see when you press **CTRL-F**. Field Service personnel will then give you further instructions for the appropriate action to be taken.

An example of the I/O Fault Zoom screen displaying this information is shown below.



```
0.6                %Q  00033   LOSS OF I/O MODULE                    04-24 13:21:36
02 482100 00067F7FFF7F 0303 0E 00 00 020000000000000000000000000000000000000000
                          Loss of I/O Module

The PLC operating software generates this error when it detects that a
Model 70 I/O module is no longer responding to commands from the PLC CPU,
or when the configuration file indicates an I/O module is to occupy a slot
and no module exists in the slot.

Corrective Action
-----------------
1.) Replace the module.
2.) Correct the configuration file.
3.) Display the PLC fault table on the programmer. Contact GE Fanuc PLC
    Field Service, giving them all the information contained in the fault
    entry.
```

Section 3: PLC Fault Table Explanations

Each fault explanation contains a fault description and instructions to correct the fault. Many fault descriptions have multiple causes. In these cases, the error code, displayed with the additional fault information obtained by pressing **CTRL-F**, is used to distinguish different fault conditions sharing the same fault description. The error code is the first two hexadecimal digits in the fifth group of numbers, as shown in the following example.

```
01    000000    01030100    0902    0200    000000000000
                                    └─ Error Code (first two hex
                                       digits in fifth group
```

Some faults can occur because random access memory on either the PLC CPU board or the expansion memory board has failed. These same faults may also occur because the system has been powered off and the battery voltage is (or was) too low to maintain memory. To avoid excessive duplication of instructions when corrupted memory may be a cause of the error, the correction simply states:

`Perform the corrections for Corrupted Memory.`

This means:

1. If the system has been powered off, replace the battery. Battery voltage may be insufficient to maintain memory contents.
2. Replace the expansion memory board. Integrated circuits on the memory board may be failing.
3. Replace the PLC CPU board. The integrated circuits on the PLC CPU board may be failing.

The following table enables you to quickly find a particular fault explanation in this section. Each entry is listed as it appears on the programmer screen.

Fault Description	Page
Loss of or Missing Rack	3-20
Loss of or Missing Option Module	3-21
Addition of or Extra Rack	3-23
Reset of, Addition of, or Extra Option Module	3-23
System Configuration Mismatch	3-24
System Bus Error	3-27
PLC CPU Hardware Failure	3-27
Module Hardware Failure	3-28
Option Module Software Failure	3-29
Program or Block Checksum Failure	3-30
Low Battery Signal	3-30
Constant Sweep Time Exceeded	3-31
PLC System Fault Table Full	3-31
I/O Fault Table Full	3-31
Application Fault	3-32
Non-Configurable Faults	3-33
System Bus Failure	3-33
No User Program on Power-Up	3-34
Corrupted User Program on Power-Up	3-34
Window Completion Failure	3-35
Password Access Failure	3-35
Null System Configuration for **RUN** Mode	3-35
PLC CPU System Software Failure	3-36
Too Many Bus Controllers	3-37
Communications Failure During Store	3-38
RUN MODE STORE Failure	3-38

Note

For information about values for fault groups, refer to Appendix B.

Configurable Faults

Configurable faults can have their fault action (fatal or diagnostic) changed. The CPU uses the fault action specified by the configurator utility; this may be the default action or a fault action chosen by the user. In this section, the default fault action is listed for configurable faults.

Loss of or Missing Rack

The Fault Group **`Loss of or Missing Rack`** occurs when the system cannot communicate with an expansion rack because the BTM in the main rack failed, the BRM in the expansion rack failed, power failed in the expansion rack or the expansion rack was configured in the configuration file but did not respond during power-up. The default fault action for this group is **`diagnostic`**.

Error Code:	1
Name:	Rack Lost
Description:	The PLC operating software (System Configurer) generates this error when the main rack can no longer communicate with an expansion rack. The error is generated for each expansion rack that exists in the system.
Correction:	(1) Power off the system. Verify that both the BTM and the BRM are seated properly in their respective racks and that all cables are properly connected and seated. (2) Replace the cables. (3) Replace the BRM. (4) Replace the BTM.
Error Code:	2
Name:	Rack Not Responding
Description:	The PLC operating software (System Configurer) generates this error when the configuration file is stored from the programmer indicates that a particular expansion rack should be in the system, but none responds for that rack number.
Correction:	(1) Check rack number jumper behind power supply, first on missing rack and then on all other racks, for duplicated rack numbers. (2) Update the configuration file if a rack should not be present. (3) Add the rack to the hardware configuration if a rack should be present and one is not. (4) Power off the system. Verify that both the BTM and the BRM are seated properly in their respective racks and that all cables are properly connected and seated. (5) Replace the cables. (6) Replace the BRM. (7) Replace the BTM.

Loss of or Missing Option Module

The Fault Group **Loss of or Missing Option Module** occurs when a GEnet, PCM, BTM, or BRM fails to respond. The failure may occur at power-up if the module is missing or during operation if the module fails to respond. The default fault action for this group is **Diagnostic**.

Error Code:	3
Name:	Bus Transmitter Module Found in Expansion Rack
Description:	The PLC operating software (System Configurer) generates this error when a Bus Transmitter Module is found in an expansion rack.
Correction:	Power off the system and remove the BTM from the expansion rack.
Error Code:	16
Name:	Analog Expander located to the left of the Base Converter module.
Description:	An Analog Expander module has been placed in a rack to the left of its Base Converter module.
Correction:	Power off the system. Move the Analog Expander module to the right of the Base Converter module.
Error Code:	19
Name:	Lost Analog Expander module
Description:	Base Converter module has lost communications with the Analog Expander module.
Correction:	(1) Verify wiring linking Base Converter module with the Analog Expander module. (2) Replace the Analog Expander module. (3) If all Analog Expanders lost, replace the Base Converter module.
Error Code:	2C, 2D
Name:	Option Module Soft Reset Failed
Description:	PLC CPU unable to re-establish communications with option module after soft reset.
Correction:	(1) Try soft reset a second time. (2) Replace the option module. (3) Power off the system. Verify that both the BTM and BRM are seated properly in their respective racks and that all cables are properly connected and seated. (4) Replace the cables. (5) Replace the BRM. (6) Replace the BTM. (7) Report failure to GE Fanuc PLC Field Service.
Error Code:	3B
Name:	Loss of, or missing communications driver
Description:	The PLC operating software generates this error when VME communications fail between the PLC CPU and a third party VME module using the FULL MAIL configuration mode.
Correction:	(1) Update the configuration file with the correct communications parameters. (2) Replace the communications driver on the module. (3) Remove the module from the configuration file. (4) Replace the module.

Error Code:	3C
Name:	Module in firmware update mode
Description:	The PLC operating software (System Configurer) generates this error when it finds a module in firmware update mode. Modules in this mode will not communicate with the PLC CPU.
Correction:	(1) Run the firmware update utility for the module. (2) Reset the module with the push-button. (3) Power-cycle the entire system. (4) Power-cycle the rack containing the module.
Error Code:	41
Name:	Unable to establish VME communications
Description:	The PLC operating software (System Configurer) generates this error when it finds a module in standalone mode. A module in standalone mode will appear to be operating correctly, but it will not communicate with the PLC CPU.
Correction:	(1) Reset the module with the pushbutton. (2) Power-cycle the entire system. (3) Power-cycle the rack containing the module.
Error Code:	FF
Name:	Option Module Communications Failed
Description:	PLC CPU generates this error when communications to the option module has failed.
Correction:	(1) Check the bus for abnormal activity. (2) Replace the intelligent option module to which the request was directed. (3) Check the parallel programmer cable for proper attachment.
Error Code:	All Others
Name:	Module Failure During Configuration
Description:	The PLC operating software (CPU VME Communications) generates this error when a module fails during power-up or configuration store.
Correction:	(1) Power off the system. Replace the module located in that rack and slot. (2) If the board is located in an expansion rack, verify BTM/BRM cable connections are tight and the modules are seated properly; verify the addressing of the expansion rack. (3) Replace the BTM. (4) Replace the BRM. (5) Replace the rack.

Addition of or Extra Rack

The Fault Group **Addition of or Extra Rack** occurs when a configured expansion rack with which the PLC CPU could not communicate comes online or is powered on, or an unconfigured rack is found. The default fault action for this group is **Diagnostic**.

Error Code:	1
Name:	Extra Rack
Correction:	(1) Check rack jumper behind power supply for correct setting. (2) Update the configuration file to include the expansion rack. (3) Remove the expansion rack from the hardware configuration. Note:: No correction necessary if rack was just powered on.

Reset of, Addition of, or Extra Option Module

The Fault Group **Reset of, Addition of, or Extra Option Module** occurs when an option module (PCM, BTM, etc.) comes online, is reset, or a module is found in the rack but none is specified in the configuration. The default fault action for this group is **Diagnostic**.

Error Code:	1
Name:	Extra Option Module
Correction:	(1) Update the configuration file to include the module. (2) Remove the module from the system.
Error Code:	2
Name:	Module Restart Complete
Description:	Restart of module is complete.
Correction:	None
Error Code:	3
Name:	LAN Interface Restart Complete, Running Utility
Description:	The LAN Interface module has restarted and is running a utility program.
Correction:	Refer to the LAN Interface manual, GFK-0868 or GFK-0869 (previously GFK-0533).

System Configuration Mismatch

The Fault Group **Configuration Mismatch** occurs when the module occupying a slot is different from that specified in the configuration file. The default fault action is **Fatal**. When the I/O Scanner generates the mismatch because of a Genius block, the second byte in the *Fault Extra Data* field contains the bus address of the mismatched block.

Error Code:	2
Name:	Genius I/O Block Number Mismatch
Description:	The PLC operating software (I/O Scanner) generates this fault when the configured and physical Genius I/O blocks have different model numbers.
Correction:	(1) Replace the Genius I/O block with one corresponding to configured module. (2) Update the configuration file.
Error Code:	4
Name:	I/O Type Mismatch
Description:	The PLC operating software (I/O Scanner) generates this fault when the physical and configured I/O types of Genius grouped blocks are different.
Correction:	(1) Remove the indicated Genius module and install the module indicated in the configuration file. (2) Update the Genius module descriptions in the configuration file to agree with what is physically installed.
Error Code:	7
Name:	Daughter Board Mismatch
Description:	The PLC operating software (Service Request Processor) generates this error when the configuration file indicates one size memory daughter (expansion) board should be on the PLC CPU and a different size is actually present.
Correction:	(1) Replace the module. (2) Replace the daughter board with the size indicated in the configuration file. (3) Update the configuration file to agree with the size of the daughter board actually installed on the PLC CPU.
Error Code:	8
Name:	Analog Expander Mismatch
Description:	The PLC operating software (Service Request Processor) generates this error when the configured and physical Analog Expander modules have different model numbers.
Correction:	(1) Replace the Analog Expander module with one corresponding to configured module. (2) Update the configuration file.
Error Code:	9
Name:	Genius I/O Block Size Mismatch
Description:	The PLC operating software (Service Request Processor) generates this error when block configuration size does not match the configured size.
Correction:	Reconfigure the block.

Error Code:	A
Name:	Unsupported Feature
Description:	Configured feature not supported by this revision of the module.
Correction:	(1) Update the module to a revision that supports the feature. (2) Change the module configuration.
Error Code:	B
Name:	Revision A of BTM Not in Right-Most Slot
Description:	The BTM (Revision A version) is not the right-most module in the rack.
Correction:	(1) Move the BTM to the right of all other modules in the rack. (2) Upgrade the BTM to a newer version (Revision B, or higher).
Error Code:	E
Name:	LAN Duplicate MAC Address
Description:	This LAN Interface module has the same MAC address as another device on the LAN. The module is off the network.
Correction:	(1) Change the module's MAC address. (2) Change the other device's MAC address.
Error Code:	F
Name:	LAN Duplicate MAC Address Resolved
Description:	Previous duplicate MAC address has been resolved. The module is back on the network. This is an informational message.
Correction:	None required.
Error Code:	10
Name:	LAN MAC Address Mismatch
Description:	MAC address programmed by softswitch utility does not match configuration stored from Logicmaster 90 software.
Correction:	Change MAC address on softswitch utility or in Logicmaster 90 configuration software.
Error Code:	11
Name:	LAN Softswitch/Modem mismatch
Description:	Configuration of LAN module does not match modem type or configuration programmed by softswitch utility.
Correction:	(1) Correct configuration of modem type. (2) Consult LAN Interface manual for configuration setup.
Error Code:	17
Name:	Invalid Memory Reference
Description:	Memory references in the logic program exceed that which is available.
Correction:	Update the configuration file and store it to the PLC.
Error Code:	1E
Name:	Reference length mismatch
Description:	The PLC operating software (I/O Scanner) generates this error when the I/O reference lengths specified in the configuration for this module do not match the actual data sizes reported by the board.
Correction:	Update the configuration file with the correct reference lengths.
Error Code:	1F
Name:	Invalid configuration parameters
Description:	The PLC operating software (System Configurer) generates this error when it determines that critical values in the module's configuration are unacceptable.
Correction:	Update the configuration file with the correct values.

Error Code:	20
Name:	New configuration requires reset
Description:	The PLC operating software (System Configurer) generates this error when it determines that a store of configuration attempted to change critical configuration values for the specified module. The new configuration will not take effect until the module is reset.
Correction:	(1) Power-cycle the entire system. (2) Power-cycle the rack containing the module.
Error Code:	27
Name:	Unresolved or disabled interrupt reference
Description:	The PLC operating software (I/O Scanner) generates this error when an interrupt trigger reference is either out of range or disabled in the the I/O module's configuration.
Correction:	(1) Remove or correct the interrupt trigger reference. (2) Update the configuration file to enable this particular interrupt.
Error Code:	1D
Name:	Incompatible scheduling mode
Description:	A program with a scheduling mode that is incompatible with the sweep mode has been stored. Logged on a stop-to-run transition.
Correction:	(1) Change the sweep mode and try again. (2) Change the scheduling mode or delete the offending program(s) from the program declaration screen.
Error Code:	24
Name:	I/O specification mismatch
Description:	The I/O specification of a program does not match the specification given in the program.
Correction:	Correct the mismatch between the I/O specification by changing the I/O specification declaration, or corresponding macro declaration in the C program source file.
Error Code:	25
Name:	Controller reference out of range
Description:	A reference on either the trigger, disable, or I/O specification is out of the configured limits.
Correction:	Modify the incorrect reference to be within range, or increase the configured size of reference data.
Error Code:	26
Name:	Bad program specification
Description:	The I/O specification of a program is corrupted.
Correction:	Contact GE Fanuc Field Service.
Error Code:	All Others
Name:	Module and Configuration Do Not Match
Description:	The PLC operating software (System Configurer) generates this fault when the module occupying a slot is not of the same type that the configuration file indicates should be in that slot.
Correction:	(1) Replace the module in the slot with one of the type that the configuration file indicates is in that slot. (2) Update the configuration file.

System Bus Error

The Fault Group **System Bus Error** occurs when the PLC CPU receives the non-configurable interrupt bus error from the bus system. The default fault action is **Diagnostic**.

Error Code:	4
Name:	Unrecognized VME Interrupt Source
Description:	The PLC operating software (Operating System) generates this error when a module generates an interrupt not expected by the CPU (unconfigured or unrecognized).
Correction:	(1) Ensure that all modules configured for interrupts have corresponding interrupt declarations in the program logic. (2) Ensure that no third-party VME module is generating interrupts on the IRQ6 and IRQ7 lines.
Error Code:	All Others
Name:	System Bus Error
Description:	The PLC operating software (Operating System) generates this fault when it has detected an error signal on the VME backplane, such as a parity error.
Correction:	(1) Ensure that all expansion rack cables are properly connected and seated. (2) Take action to minimize system noise.

PLC CPU Hardware Failure

The Fault Group **PLC CPU Hardware** occurs when the PLC CPU detects a hardware failure, such as a RAM failure or a communications port failure. When the failure is a RAM failure, the address of the failure is stored in the first four bytes of the ***Fault Extra Data*** field.

When a PLC CPU Hardware failure occurs, the PLC OK LED will flash on and off to indicate that the failure was not serious enough to prevent programmer communications to retrieve the fault information. The default fault action for this group is **Fatal**.

Error Code:	6Eh
Name:	Time-of-Day Clock Not Battery-Backed
Description:	The battery-backed value of the time-of-day clock has been lost.
Correction:	(1) Replace the battery. Do not remove power from the main rack until replacement is complete. Reset the time-of-day clock using Logicmaster 90 software. (2) Replace the module.
Error Code:	All Others
Correction:	Replace the module.

Module Hardware Failure

The Fault Group **`Module Hardware Failure`** occurs when the PLC CPU detects non-fatal hardware failure on any module in the system, e.g., a serial port failure on a PCM. The default fault action for this group is **`Diagnostic`**.

Error Code:	1A0
Name:	Missing 12 Volt Power Supply
Description:	A power supply that supplies 12 volts is required to operate the LAN Interface module.
Correction:	(1) Install/replace a GE Fanuc 100 watt power supply. (2) Connect an external VME power supply that supplies 12 volts.
Error Code:	1C2 - 1C6
Name:	LAN Interface Hardware Failure
Description:	Refer to the LAN Interface manual, GFK-0868 or GFK-0869 (previously GFK-0533), for a description of these errors.
Error Code:	All Others
Name:	Module Hardware Failure
Description:	A module hardware failure has been detected.
Correction:	Replace the affected module.

Option Module Software Failure

The Fault Group **Option Module Software Failure** occurs when a non-recoverable software failure occurs on a PCM. It is also generated when the identification data read from a module indicates that the module is a GE Fanuc module but the module type is not a supported GE Fanuc type. The default fault action for this group is **Fatal**.

Error Code:	1
Name:	Unsupported Board Type
Description:	The PLC operating software (System Configurer) generates this fault when the identification data read from a board indicates that the board is a GE Fanuc board but the type of board is not one of the GE Fanuc board types.
Correction:	(1) Upload the configuration file and verify that the software recognizes the board type in the file. If there is an error, correct it, download the corrected configuration file and retry. (2) Display the PLC fault table on the programmer. Contact GE Fanuc PLC Field Service, giving them all the information contained in the fault entry.
Error Code:	2, 3
Name:	COMMREQ Frequency Too High
Description:	COMMREQs are being sent to a module faster than it can process them.
Correction:	Change the PLC program to send COMMREQs to the affected module at a slower rate, or monitor the completion status of each COMMREQ before sending the next.
Error Code:	4
Name:	More Than One BTM in a Rack
Description:	There is more than one BTM present in the rack.
Correction:	Remove one of the BTMs from the rack; there can only be one in a CPU rack.
Error Code:	191, 195
Name:	LAN Interface Software Failure
Description:	Refer to the LAN Interface manual, GFK-0868 or GFK-0869 (previously GFK-0533), for a description of these errors.
Error Code:	All Others
Name:	Option Module Software Failure
Description:	Software failure detected on an option module.
Correction:	(1) Reload software into the indicated module. (2) Replace the module.

Program or Block Checksum Failure

The Fault Group **`Program or Block Checksum Failure`** occurs when the PLC CPU detects error conditions in program or blocks received by the PLC. It also occurs during **`RUN`** mode background checking. In all cases, the *Fault Extra Data* field of the PLC fault table record contains the name of the program or block in which the error occurred. The default fault action for this group is **`Fatal`**.

Error Code:	All
Name:	Program or Block Checksum Failure
Description:	The PLC operating software generates this error when a program or block is corrupted.
Correction:	(1) Clear PLC memory and retry the store. (2) Examine C application for errors. (3) Display the PLC fault table on the programmer. Contact GE Fanuc PLC Field Service, giving them all the information contained in the fault entry.

Low Battery Signal

The Fault Group **`Low Battery Signal`** occurs when the PLC CPU detects a low battery on the PLC CPU board, the PLC CPU memory daughter board, or a module such as the PCM reports a low battery condition. The default fault action for this group is **`Diagnostic`**.

Error Code:	0
Name:	Failed Battery Signal
Description:	The CPU module (or other module having a battery) battery is dead.
Correction:	Replace the battery. Do not remove power from the rack until replacement is complete.
Error Code:	1
Name:	Low Battery Signal
Description:	A battery on the CPU or other module has a low signal.
Correction:	Replace the battery. Do not remove power from the rack until replacement is complete.

Constant Sweep or Microcycle Time Exceeded

The Fault Group **`Constant Sweep or Microcycle Time Exceeded`** occurs when the PLC CPU operates in **`CONSTANT SWEEP`** or **`MICROCYCLE SWEEP`** mode and it detects that the sweep has exceeded the constant sweep timer. The fault extra data contains the name of the folder in eight bytes. The default fault action for this group is **`Diagnostic`**.

Error Code:	0 Constant Sweep 1 Microcycle
Correction:	(1) Increase constant sweep time. (2) Remove logic from application program. If Microcycle: (1) Increase the time. (2) Modify execution intervals to give programs more time to execute.

PLC System Fault Table Full

The Fault Group **`PLC System Fault Table Full`** occurs when the PLC CPU places the 16th entry in the PLC fault table. The default fault action for this group is **`Diagnostic`**.

Error Code:	0
Correction:	Clear the PLC fault table.

I/O Fault Table Full

The Fault Group **`I/O Fault Table Full`** occurs when the PLC CPU places the 32nd entry in the I/O fault table. To avoid loss of additional faults, clear the earliest entry from the table. The default fault action for this group is **`Diagnostic`**.

Error Code:	0
Correction:	Clear the I/O fault table.

Application Fault

The Fault Group **Application Fault** occurs when the PLC CPU detects a fault in the user program. The default fault action for this group is **Diagnostic**.

Error Code:	1
Name:	Indirect Address Out of Range
Description:	The PLC operating software generates this error when one of the parameters to a function block is an indirect reference (i.e., the parameter is an address within that memory type which contains the parameter value) and the contents of the indirect reference are out of range for the memory type. For example, consider a system with 500 %R registers defined. This fault would be generated if the parameter address were %R00100, and the contents of %R00100 were greater than 500 or zero. The Fault Extra Data field contains in the first two bytes the offset address of where the call was made, the segment selector and offset (reference) in the next four bytes, and the name of the program or block in which the function call resides in the next eight bytes.
Correction:	(1) Correct the indirect reference. (2) Increase the number of registers available, if possible.
Error Code:	2
Name:	Software Watchdog Timer Expired
Description:	The PLC operating software generates this error when the watchdog timer expires. The PLC CPU stops executing the user program and enters **STOP** mode. The only recovery is to cycle power to the PLC CPU. Examples causing timer expiration: Looping, via jump, very long program, etc.
Correction:	(1) Determine what caused the expiration (logic execution, external event, etc.) and correct. (2) Use the system service function block to restart the watchdog timer.
Error Code:	5
Name:	COMMREQ **WAIT** Mode Not Supported
Description:	The module receiving the COMMREQ does not support **WAIT** mode COMMREQs.
Correction:	Use **NOWAIT** mode COMMREQs.
Error Code:	6
Name:	COMMREQ Bad Task ID
Description:	The task selected by the COMMREQ does not exist on the option module.
Correction:	Correct the task ID.
Error Code:	7
Name:	Application Stack Overflow
Description:	Block call depth has exceeded the PLC capability.
Correction:	Increase the program's stack size or adjust application program to reduce nesting.
Error Code:	8 through D
Name:	LAN Interface Application Faults
Description:	Refer to the LAN Interface manual, GFK-0868 or GFK-0869 (previously GFK-0533), for a description of these errors.
Error Code:	1C
Name:	Program Exceeded Wind Down
Description:	A program failed to complete execution within the wind-down period (currently 2.5 seconds) after the PLC was commanded to STOP.
Correction:	A program has gone into an infinite loop or is taking too long to execute. Correct the coding error or modify the program.

Error Code:	1D
Name:	Program Not Readied
Description:	A program scheduled to be readied has not completed its previous execution. The base cycle time is too small (Periodic programs), or the interrupt rate is too high (Event-triggered or Timed programs).
Correction:	(1) Increase the base cycle time, or decrease the interrupt rate. (2) Increase the execution interval time to allow the program to finish execution.
Error Code:	0E
Name:	External Block Run-Time Error
Description:	A run-time error occurred during execution of an external block.
Correction:	Based on the fault information, correct the specific problem in the external block.
Error Code:	0F
Name:	SORT Interrupt Error
Description:	A SORT function executed in a timed or I/O interrupt at the same time a SORT function was executing in another block.
Correction:	Do not use the SORT function in both interrupt and non-interrupt blocks.
Error Code:	11
Name:	Standalone Run-Time Error
Description:	A run-time error occurred during execution of a Standalone program.
Correction:	Based on the fault information, correct the specific problem in the external block.

Non-Configurable Faults

The fault action of **Non-Configurable Faults** cannot be changed. **Fatal** faults cause the PLC to enter a form of **STOP** mode at the end of the sweep in which the error occurred. **Diagnostic** faults are logged and corresponding fault contacts are set. **Informational** faults are simply logged in the PLC fault table.

System Bus Failure

The Fault Group **System Bus Failure** occurs when the PLC CPU software receives the non-configurable interrupt bus failure from the bus system. The default fault action for this group is **Fatal**.

Error Code:	1
Name:	Bus Grant Failure
Description:	The PLC operating software generates this error when the PLC CPU was unable to obtain control of the VME bus when required.
Correction:	(1) Ensure that any non-GE Fanuc boards which can become bus masters are relinquishing control of the VME bus when requested to do so by the PLC CPU. (2) Replace the PLC CPU module.

No User Program on Power-Up

The Fault Group **No User Program on Power-Up** occurs when the PLC CPU powers up with its memory preserved but no user program exists in the PLC. The PLC CPU detects the absence of a user program on power-up; the controller stays in **STOP** mode, performing the **STOP** mode sweep until a valid program is downloaded. The default fault action for this group is **Informational**.

Correction:	Download an application program before attempting to go to **RUN** mode.

Corrupted User Program on Power-Up

The Fault Group **Corrupted User Program on Power-Up** occurs when the PLC CPU detects corrupted user RAM. The PLC CPU will remain in **STOP** mode until a valid user program and configuration file are downloaded. The default fault action for this group is **Fatal**.

Error Code:	1
Name:	Corrupted User RAM on Power-Up
Description:	The PLC operating software (Operating Software) generates this error when it detects corrupted user RAM on power-up.
Correction:	(1) Reload the configuration file, user program, and references (if any). (2) Examine any C applications for errors. (3) Replace the battery on the PLC CPU. (4) Replace the expansion memory board on the PLC CPU. (5) Replace the PLC CPU.
Error Code:	2
Name:	Illegal Boolean OpCode Detected
Description:	The PLC operating software (Operating Software) generates this error when it detects a bad instruction in the user program.
Correction:	(1) Restore the user program and references, if any. (2) Examine any C applications for errors. (3) Replace the expansion memory board on the PLC CPU. (4) Replace the PLC CPU.
Error Code:	6
Name:	Corrupted Remote I/O Scanner EEPROM
Description:	The configuration in the Remote I/O Scanner EEPROM was found to be corrupted at power-up.
Correction:	Restore the Remote I/O Scanner configuration.

Window Completion Failure

The Fault Group **`Window Completion Failure`** is generated by the pre-logic and end-of-sweep processing software in the PLC. The fault extra data contains the name of the task that was executing when the error occurred. The default fault action for this group is **`Informational`**.

Error Code:	0
Name:	Window Completion Failure
Description:	The PLC operating software generates this error when the PLC is operating in **`CONSTANT SWEEP`** mode and the constant sweep time was exceeded before the programmer window had a chance to begin executing.
Correction:	Increase the constant sweep timer value.
Error Code:	1
Name:	Logic Window Skipped
Description:	The logic window was skipped due to lack of time to execute.
Correction:	(1) Increase base cycle time. (2) Reduce Communications window time.

Password Access Failure

The Fault Group **`Password Access Failure`** occurs when the PLC CPU receives a request to change to a new privilege level and the password included with the request is not valid for that level. The default fault action for this group is **`Informational`**.

Error Code:	0
Correction:	Retry the request with the correct password.

Null System Configuration for RUN Mode

The Fault Group **`Null System Configuration for RUN Mode`** occurs when the PLC transitions from **`STOP`** to one of the **`RUN`** modes and a configuration file is not present. The transition to **`RUN`** is permitted, but no I/O scans occur. The effect of this fault is to perform the function of a Suspend I/O. The default fault action for this group is **`Informational`**.

Error Code:	0
Correction:	Download a configuration file.

PLC CPU System Software Failure

Faults in the Fault Group **PLC CPU System Software Failure** are generated by the operating software of the Series 90-70 PLC CPU. They occur at many different points of system operation. When a **Fatal** fault occurs, the PLC CPU immediately transitions into a special **ERROR SWEEP** mode. The only activity permitted when the PLC is in this mode is communications with the programmer. The only method of clearing this condition is to cycle power on the PLC. The default fault action for this group is **Fatal**.

Error Code:	14, 27
Name:	Corrupted PLC Program Memory
Description:	The PLC operating software generates these errors when certain PLC operating software problems occur. These should *not* occur in a production system.
Correction:	(1) Display the PLC fault table on the programmer. Contact GE Fanuc PLC Field Service, giving them all the information contained in the fault entry. (2) Perform the corrections for corrupted memory.
Error Code:	52
Name:	Backplane Communications Failed
Description:	The PLC operating software (Service Request Processor) generates this error when it attempts to comply with a request that requires backplane communications and receives a rejected mail response.
Correction:	(1) Check the bus for abnormal activity. (2) Replace the intelligent option module to which the request was directed. (3) Check parallel programmer cable for proper attachment.

Error Code:	5A
Name:	User Shut Down Requested
Description:	The PLC operating software (function blocks) generates this informational alarm when SVCREQ #13 (User Shut Down) executes in the application program.
Correction:	None required. Information-only alarm.
Error Code:	7B
Name:	Remote I/O Scanner Communications Heartbeat Failure
Description:	Refer to the *Series 90-70 Remote I/O Scanner User's Manual,* GFK-0579, for a description of this error.
Correction:	None required. Information-only alarm.
Error Code:	94
Name:	Units Contain Mismatched Firmware, Update Recommended
Description:	This fault is logged each time the redundancy state changes and the two 780 CPUs contain incompatible firmware.
Correction:	Ensure that both 780 CPUs have compatible firmware.
Error Code:	All Others
Name:	PLC CPU Internal System Error
Description:	An internal system error has occurred that should **not** occur in a production system.
Correction:	Display the PLC fault table on the programmer. Contact GE Fanuc PLC Field Service, giving them all the information contained in the fault entry.

Too Many Bus Controllers

The Fault Group **`Too Many Bus Controllers`** occurs when the the I/O Scanner portion of the PLC operating software detects that more than the maximum number (32) of bus controllers have been defined. The PLC CPU itself is a bus controller for the Model 70 I/O present in the system. The default fault action for this group is **`Fatal`**.

Note

Genius bus controllers which are configured for redundant and non-redundant blocks count as two bus controllers.

Correction:	(1) Determine which modules are bus controllers and remove the extra ones. (2) Delete a bus controller from the configuration file and store the file to the PLC CPU. (3) If bus controllers have been moved from one slot in the rack to a different slot and this error did not occur before the move, cycle power on the rack. **No module should be inserted with power applied to rack.** (4) Display the PLC fault table on the programmer. Contact GE Fanuc PLC Field Service, giving them all the information contained in the fault entry.

Communications Failure During Store

The Fault Group **Communications Failure During Store** occurs during the store of programs or blocks and other data to the PLC. The stream of commands and data for storing programs or blocks and data starts with a special start-of-sequence command and terminates with an end-of-sequence command. If communications with the programming device performing the store is interrupted or any other failure occurs which terminates the store, this fault is logged. As long as this fault is present in the system, the controller will not transition to **RUN** mode.

This fault is *not* automatically cleared on power-up; the user must specifically clear the condition. The default fault action for this group is **Fatal**.

Error Code:	0
Correction:	Clear the fault and retry the download of the program or configuration file.

Run Mode Store Failure

Error Code:	1
Description:	Communications was lost, or power was lost during a **RUN MODE STORE**. The new program or block was not activated and was deleted.
Correction:	Perform the **RUN MODE STORE** again. This fault is diagnostic.
Error Code:	2
Description:	Communications was lost, or power was lost during the cleanup of old programs or blocks during a **RUN MODE STORE**. The new program or block is installed, and the remaining programs and blocks were cleaned up.
Correction:	None required. This fault is informational.
Error Code:	3
Description:	Power was lost in the middle of a **RUN MODE STORE**.
Correction:	Delete and restore the program. This error is fatal.

Section 4: I/O Fault Table Explanations

The I/O fault table reports data about faults in three classifications. In descending order, these fault classifications are:

- Fault category.
- Fault type.
- Fault description.

All faults have a fault category, but a fault type and fault group may not be listed for every fault. Additional information, referred to as Fault Specific Data, can be accessed by pressing **CTRL-F** while the I/O fault table is displayed on the programmer screen. This section is organized by fault category.

The following table enables you to quickly find a particular fault explanation in this section. Each entry is listed as it appears on the programmer screen.

Fault Description	Page
Circuit Fault	3-40, 3-42
Discrete Fault	3-40, 3-43
Analog Fault	3-40, 3-44
Low-Level Analog Fault	3-40, 3-45
GENA Fault	3-40, 3-46
Loss of IOC	3-40, 3-46
Addition of IOC	3-40, 3-46
Loss of I/O Module	3-40, 3-47
Addition of I/O Module	3-40, 3-47
Extra I/O Module	3-40, 3-47
Loss of Block	3-40, 3-48
Addition of Block	3-40, 3-48
Extra Block	3-40, 3-48
I/O Bus Fault	3-40, 3-49
Module Fault	3-41, 3-50
IOC Software Fault	3-41, 3-50
IOC Hardware Failure	3-41, 3-51
Forced and Unforced Circuit	3-41, 3-51
Block Switch	3-41, 3-51

The following table describes the information provided with each fault category.

Table 3-6. Fault Category Descriptions

Fault Category	Fault Type	Fault Description	Fault Specific Data
Circuit Fault	Discrete Fault	Loss of User Side Power	Circuit Configuration *
		Short Circuit in User Wiring	Circuit Configuration *
		Sustained Overcurrent	Circuit Configuration *
		Low or No Current Flow	Circuit Configuration *
		Switch Temperature Too High	Circuit Configuration *
		Switch Failure	Circuit Configuration *
		Point Fault	Circuit Configuration *
		Output Fuse Blown	Circuit Configuration *
	Analog Fault	Input Channel Low Alarm	Circuit Configuration *
		Input Channel High Alarm	Circuit Configuration *
		Input Channel Under Range	Circuit Configuration *
		Input Channel Over Range	Circuit Configuration *
		Input Channel Open Wire	Circuit Configuration *
		Output Channel Under Range	Circuit Configuration *
		Output Channel Over Range	Circuit Configuration *
		Invalid Data	Circuit Configuration *
		Expansion Channel Not Responding	Circuit Configuration *
	Low-Level Analog Fault	Input Channel Low Alarm	Circuit Configuration *
		Input Channel High Alarm	Circuit Configuration *
		Input Channel Under Range	Circuit Configuration *
		Input Channel Over Range	Circuit Configuration *
		Input Channel Open Wire	Circuit Configuration *
		Wiring Error	Circuit Configuration *
		Internal Fault	Circuit Configuration *
		Input Channel Shorted	Circuit Configuration *
		Invalid Data	Circuit Configuration *
	GENA Fault	GENA Circuit Fault	GENA Fault Byte 2
	Remote I/O Scanner Fault	Remote I/O Scanner Circuit Fault	Byte 1: Circuit Type Byte 2: I/O Type
Loss of IOC			Timeout Unexpected State Unexpected Mail Status VME Bus Error
Addition of IOC			
Loss of I/O Module			
Addition of I/O Module			
Extra I/O Module			
Loss of Block	Fault Not Specified	Communications Lost	
Addition of Block			
Extra Block			
I/O Bus Fault	Bus Fault Bus Outputs Disabled		
Global Memory Fault			Subnet Group Number Global Variable Name

* Refer to table on next page.

Table 3-4. Fault Category Descriptions (cont'd)

Fault Category	Fault Type	Fault Description	Fault Specific Data
Module Fault	Headend Fault	EPROM or NVRAM Failure Calibration Memory Failure Shared Ram Failure Configuration MisMatch Watchdog Timeout Output Fuse Blown	
IOC Software Fault			
IOC Hardware Failure			
Forced Circuit			Block Configuration * Discrete/Analog Indication*
Unforced Circuit			Block Configuration * Discrete/Analog Indication*

* Refer to table below.

Three types of fault specific data occur in more than one fault category; they are block configuration, circuit configuration, and analog/discrete indication. The codings are shown in the following table.

Value	Description
	Circuit Configuration
1	Circuit is an input.
2	Circuit is an output.
3	Circuit is an output with feedback.
	Block Configuration
1	Block is configured for inputs only.
2	Block is configured for outputs only.
3	Block is configured for inputs and outputs (grouped block).
	Discrete/Analog Indication
1	Block is a discrete block.
2	Block is an analog block.

Circuit Fault

Circuit Fault has four fault types. Three of the four fault types have fault descriptions. Fault specific data is available for all faults. Circuit fault applies specifically to Genius I/O modules. The default fault action is **Diagnostic**. The following table describes the circuit fault category.

Table 3-7. Circuit Fault Category Description

Fault Category	Fault Type	Fault Description	Fault Specific Data
Circuit Fault	Discrete Fault	Loss of User Side Power	Circuit Configuration
		Short Circuit in User Wiring	Circuit Configuration
		Sustained Overcurrent	Circuit Configuration
		Low or No Current Flow	Circuit Configuration
		Switch Temperature Too High	Circuit Configuration
		Switch Failure	Circuit Configuration
		Point Fault	Circuit Configuration
		Output Fuse Blown	Circuit Configuration
	Analog Fault	Input Channel Low Alarm	Circuit Configuration
		Input Channel High Alarm	Circuit Configuration
		Input Channel Under Range	Circuit Configuration
		Input Channel Over Range	Circuit Configuration
		Input Channel Open Wire	Circuit Configuration
		Output Channel Under Range	Circuit Configuration
		Output Channel Over Range	Circuit Configuration
		Invalid Data	Circuit Configuration
		Expansion Channel Not Responding	Circuit Configuration
	Low-Level Analog Fault	Input Channel Low Alarm	Circuit Configuration
		Input Channel High Alarm	Circuit Configuration
		Input Channel Under Range	Circuit Configuration
		Input Channel Over Range	Circuit Configuration
		Input Channel Open Wire	Circuit Configuration
		Wiring Error	Circuit Configuration
		Internal Fault	Circuit Configuration
		Input Channel Shorted	Circuit Configuration
		Invalid Data	Circuit Configuration
	Remote Fault	Remote I/O Scanner Fault	

Discrete Fault

`Discrete Fault` has eight fault descriptions. More than one condition may be present in a particular reporting of the fault.

Name:	Loss of User Side Power
Description:	The Genius Bus Controller generates this error when there is a power loss on the field wiring side of a Genius I/O block.
Correction:	Correct the power failure.
Name:	Short Circuit in User Wiring
Description:	The Genius Bus Controller generates this error when it detects a short circuit in the user wiring of a Genius block. A short circuit is defined as a current level greater than 20 amps.
Correction:	Fix the cause of the short circuit.
Name:	Sustained Overcurrent
Description:	The Genius Bus Controller generates this error when it detects a sustained current level greater than 2 amps in the user wiring.
Correction:	Fix the cause of the over current.
Name:	Low or No Current Flow
Description:	The Genius Bus Controller generates this error when there is very low or no current flow in the user circuit.
Correction:	Fix the cause of the condition.
Name:	Switch Temperature Too High
Description:	The Genius Bus Controller generates this error when the Genius block reports a high temperature in the Genius Smart Switch.
Correction:	(1) Ensure that the block is installed to provide adequate circulation. (2) Decrease the ambient temperature surrounding the block.
Name:	Switch Failure
Description:	The Genius Bus Controller generates this error when the Genius block reports a failure in the Genius Smart Switch.
Correction:	Replace the Genius I/O block.
Name:	Point Fault
Description:	The PLC operating system generates this error when it detects a failure of a single I/O point on a Genius I/O module.
Correction:	Replace the Genius I/O block.
Name:	Output Fuse Blown
Description:	The PLC operating system generates this error when it detects a blown fuse on a Genius I/O output block.
Correction:	(1) Determine and repair the cause of the fuse blowing, and replace the fuse. (2) Replace the block.

Analog Fault

`Analog Fault` has nine fault descriptions. More than one condition may be present in a particular reporting of the fault.

Name:	Input Channel Low Alarm
Description:	The Genius Bus Controller generates this error when the Genius analog module reports a low alarm on an input channel.
Correction:	Correct the condition causing the low alarm.
Name:	Input Channel High Alarm
Description:	The Genius Bus Controller generates this error when the Genius analog module reports a high alarm on an input channel.
Correction:	Correct the condition causing the high alarm.
Name:	Input Channel Under Range
Description:	The Genius Bus Controller generates this error when the Genius analog module reports an under-range condition on an input channel.
Correction:	Correct the problem causing the condition.
Name:	Input Channel Over Range
Description:	The Genius Bus Controller generates this error when the Genius analog module reports an over-range condition on an input channel.
Correction:	Correct the problem causing the condition.
Name:	Input Channel Open Wire
Description:	The Genius Bus Controller generates this error when the Genius analog module detects an open wire condition on an input channel.
Correction:	Correct the problem causing the condition.
Name:	Output Channel Under Range
Description:	The Genius Bus Controller generates this error when the Genius analog module reports an under-range condition on an output channel.
Correction:	Correct the problem causing the condition.
Name:	Output Channel Over Range
Description:	The Genius Bus Controller generates this error when the Genius analog module reports an over-range condition on an output channel.
Correction:	Correct the problem causing the condition.
Name:	Invalid Data
Description:	The Genius Bus Controller generates this error when it detects invalid data from a Genius analog input block.
Correction:	Correct the problem causing the condition.
Name:	Expansion Channel Not Responding
Description:	The PLC operating software (I/O Scanner) generates this error when data from an expansion channel on a multiplexed analog input board is not responding.
Correction:	(1) Check wiring to the module. (2) Replace the module.

Low-Level Analog Fault

`Low-Level Analog Fault` has nine fault descriptions. More than one condition may be present in a particular reporting of the fault.

Name:	Input Channel Low Alarm
Description:	The Genius Bus Controller generates this error when the Genius analog module reports a low alarm on an input channel.
Correction:	Correct the condition causing the low alarm.
Name:	Input Channel High Alarm
Description:	The Genius Bus Controller generates this error when the Genius analog module reports a high alarm on an input channel.
Correction:	Correct the condition causing the high alarm.
Name:	Input Channel Under Range
Description:	The Genius Bus Controller generates this error when the Genius analog module reports an under-range condition on an input channel.
Correction:	Correct the problem causing the condition.
Name:	Input Channel Over Range
Description:	The Genius Bus Controller generates this error when the Genius analog module reports an over-range condition on an input channel.
Correction:	Correct the problem causing the condition.
Name:	Input Channel Open Wire
Description:	The Genius Bus Controller generates this error when the Genius analog module detects an open wire condition on an input channel.
Correction:	Correct the problem causing the condition.
Name:	Wiring Error
Description:	The Genius Bus Controller generates this error when the Genius analog module detects an improper RTD connections or thermocouple reverse junction fault.
Correction:	Correct the problem causing the condition.
Name:	Internal Fault
Description:	The Genius Bus Controller generates this error when the Genius analog module reports a cold junction sensor fault on a thermocouple block or an internal error in an RTD block.
Correction:	Correct the problem causing the condition.
Name:	Input Channel Shorted
Description:	The Genius Bus Controller generates this error when it detects an input channel shorted on a Genius RTD or Strain Gauge Block.
Correction:	Correct the problem causing the condition.
Name:	Invalid Data
Description:	The Genius Bus Controller generates this error when it detects invalid data from a Genius analog input block.
Correction:	Correct the problem causing the condition.

GENA Fault

The **`GENA Fault`** has no fault descriptions associated with it. GENA Fault Byte 2 is the first byte of the fault specific data.

Description:	The Genius I/O operating software generates this error when it detects a failure in a GENA block attached to the Genius I/O bus.
Correction:	Replace the GENA block.

Loss of IOC

The Fault Category **`Loss of IOC`** has no fault types or fault descriptions associated with it. The default fault action is **`Fatal`**.

Name:	Loss of or Missing IOC
Description:	The PLC operating software generates this error when it cannot communicate with an I/O Controller and an entry for the IOC exists in the configuration file.
Correction:	(1) Verify that the module in the slot/bus address is the correct module. (2) Review the configuration file and verify that it is correct. (3) Replace the module. (4) Display the PLC fault table on the programmer. Contact GE Fanuc PLC Field Service, giving them all the information contained in the fault entry.

Addition of IOC

The Fault Category **`Addition of IOC`** has no fault types or fault descriptions associated with it. The default fault action for this category is **`Diagnostic`**.

Name:	Addition of IOC
Description:	The PLC operating software generates this error when an IOC which has been faulted returns to operation or when an IOC is found in the system and the configuration file indicates that no IOC is to be in that slot.
Correction:	(1) No action is necessary if faulted module is in a remote rack and is returning due to a remote rack power cycle. (2) Update the configuration file or remove the module.

Loss of I/O Module

The Fault Category `Loss of I/O Module` applies to Model 70 I/O discrete and analog modules. There are no fault types or fault descriptions associated with this category. The default fault action is `Diagnostic`.

Name:	Loss of I/O Module
Description:	The PLC operating software generates this error when it detects that a Model 70 I/O module is no longer responding to commands from the PLC CPU, or when the configuration file indicates an I/O module is to occupy a slot and no module exists in the slot.
Correction:	(1) Replace the module. (2) Correct the configuration file. (3) Display the PLC fault table on the programmer. Contact GE Fanuc PLC Field Service, giving them all the information contained in the fault entry.

Addition of I/O Module

The Fault Category `Addition of I/O Module` applies to Model 70 discrete and analog I/O modules. There are no fault types or fault descriptions associated with this category. The default fault action is `Diagnostic`.

Name:	Addition of I/O Module
Description:	The PLC operating software generates this error when an I/O module which had been faulted returns to operation.
Correction:	(1) No action necessary if module was removed or replaced, or remote rack was power cycled. (2) Update the configuration file or remove the module.

Extra I/O Module

The Fault Category `Extra I/O Module` applies only to Model 70 I/O modules. There are no fault types or fault descriptions associated with this category. The default fault action is `Diagnostic`.

Name:	Extra I/O Module
Description:	The PLC operating software generates this error when it detects a Model 70 I/O module in a slot which the configuration file indicates should be empty.
Correction:	(1) Remove the module. (It may be in the wrong slot.) (2) Update and restore the configuration file to include the extra module.

Loss of Block

The Fault Category **`Loss of Block`** applies to Genius blocks. There are no fault types or fault descriptions associated with this category. The default fault action is **`Diagnostic`**.

Name:	Loss of Block
Description:	The PLC operating software generates this error when it receives a Loss of Block fault from a Genius Bus Controller, but the reason for the loss is unspecified.
Correction:	(1) Verify power and wiring to the block. (2) Replace the block.
Name:	Loss of Block - A/D Communications Fault
Description:	The Genius I/O operating software generates this error when it detects a loss of communications with a Genius I/O block.
Correction:	(1) Verify power and serial bus wiring to the block. (2) Replace the block.

Addition of Block

The Fault Category **`Addition of Block`** applies only to Genius blocks. There are no fault types or fault descriptions associated with this category. The default fault action is **`Diagnostic`**.

Name:	Addition of Block
Description:	The Genius operating software generates this error when it detects that a Genius block which stopped communicating with the controller starts communicating again.
Correction:	Informational only. None required.

Extra Block

The Fault Category **`Extra Block`** applies only to Genius I/O blocks. There are no fault types or fault descriptions associated with this category. The default fault action is **`Diagnostic`**.

Name:	Extra Block
Description:	The PLC operating software generates this error when it detects a Genius I/O block on the bus at a serial bus address which the configuration file should not have a block.
Correction:	(1) Remove or reconfigure the block. (It may be at the wrong serial bus address.) (2) Update and restore the configuration file to include the extra block.

I/O Bus Fault

The Fault Category **I/O Bus Fault** has two fault types associated with it. The default fault action is **Diagnostic**.

Name:	Bus Fault
Description:	The Genius Bus Controller operating software generates this error when it detects a failure with a Genius I/O bus.
Correction:	(1) Determine the reason for the bus failure and correct it. (2) Replace the Genius Bus Controller.
Name:	Bus Outputs Disabled
Description:	The Genius Bus Controller operating software generates this error when it times out waiting for the PLC CPU to perform an I/O scan.
Correction:	(1) Replace the PLC CPU. (2) Display the PLC fault table on the programmer. Contact GE Fanuc PLC Field Service, giving them all the information contained in the fault entry.
Name:	SBA Conflict
Description:	The Genius Bus Controller detected a conflict between its serial bus address and that of another device on the bus.
Correction:	Adjust one of the conflicting serial bus addresses.

Module Fault

The Fault Category **`Module Fault`** has one fault type, headend fault, and eight fault descriptions. No fault specific data is present. The default fault action for this category is **`Diagnostic`**.

Name:	Configuration Memory Failure
Description:	The Genius Bus Controller generates this error when it detects a failure in a Genius block's EEPROM or NVRAM.
Correction:	Replace the Genius block's electronics module.
Name:	Calibration Memory Failure
Description:	The Genius Bus Controller generates this error when it detects a failure in a Genius block's calibration memory.
Correction:	Replace the Genius block's electronics module.
Name:	Shared RAM Fault
Description:	The Genius Bus Controller generates this error when it detects an error in a Genius block's shared RAM.
Correction:	Replace the Genius block's electronics module.
Name:	Watchdog Timeout
Description:	The PLC operating system generates this error when it detects that a Model 70 input module watchdog timer has expired.
Correction:	Replace the Model 70 input module.
Name:	Output Fuse Blown
Description:	The PLC operating system generates this error when it detects a blown fuse on a Model 70 output module.
Correction:	(1) Determine and repair the cause of the fuse blowing, and replace the fuse. (2) Replace the module.
Name:	Module Fault
Description:	An internal failure has been detected in a module.
Correction:	Replace the affected module.

IOC Software Fault

The Fault Category **`IOC Software Fault`** applies to any type of I/O Controller. There are no fault types or fault descriptions associated with it. The default fault action is **`Fatal`**.

Name:	Datagram Queue Full, Read/Write Queue Full
Description:	Too many datagrams or read/write requests have been sent to the Genius Bus Controller.
Correction:	Adjust the system to reduce the request rate to the Genius Bus Controller.
Name:	Response Lost
Description:	The Genius Bus Controller is unable to respond to a received datagram or read/write request.
Correction:	Adjust the system to reduce the request rate to the Genius Bus Controller.

IOC Hardware Failure

The Fault Category **IOC Hardware Failure** has no fault types or fault descriptions. The default fault action is **Diagnostic**.

Description:	The Genius operating software generates this error when it detects a hardware failure in the Bus Communication hardware or a baud rate mismatch.
Correction:	(1) Verify that the baud rate set in the configuration file for the Genius Bus Controller agrees with the baud rate programmed in every block on the bus. (2) Change the configuration file and restore it, if necessary. (3) Replace the Genius Bus Controller. (4) Selectively remove each block from the bus until the offending block is isolated and replace it.

Forced and Unforced Circuit

The Fault Categories **Forced Circuit** and **Unforced Circuit** report point conditions and therefore are not technically faults. They have no fault types or fault descriptions. These reports occur when a Genius I/O point was forced or unforced with the Hand-Held Monitor. The default fault action is **Informational**.

Fault Specific Data contains data as shown below.

Byte Number	Description
1	Block Configuration
2	Analog/Discrete Information

Block Switch

The Fault Category **Block Switch** has no fault types or fault descriptions. The default fault action is **Diagnostic**.

Name:	Block Switch
Description:	The PLC generates this error when a Genius block on redundant Genius buses switches from one bus to another.
Correction:	(1) No action is required to keep the block operating. (2) The bus that the block switched from needs to be repaired. (a) Verify the bus wiring. (b) Replace the I/O controller. (c) Replace the Bus Switching Module (BSM).

Chapter 4 Series 90-70 RLD Instruction Set

Programming consists of creating an application program for a PLC. This chapter describes the programming instructions which may be used to create ladder logic programs for the Series 90-70 programmable controller.

If the Logicmaster 90 programming software is not yet installed, please refer to the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263, for instructions. The user's manual explains how to create, transfer, edit, and print programs.

Configuration is the process of assigning logical addresses, as well as other characteristics, to the hardware modules in the system. It may be done either before or after programming, using the configuration software; however, it is recommended that configuration be done first. If that has not been done, you should refer to the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263, to decide whether it is best to begin programming at this time.

This chapter contains the following sections:

Section	Title	Description	Page
1	Relay Functions	Describes the use of contacts, coils, and links in ladder logic rungs.	4-2
2	Timers and Counters	Describes how to use on-delay, off-delay, and stopwatch-type timers, up counters, and down counters.	4-13
3	Math Functions	Describes the math functions of addition, subtraction, multiplication, division, modulo division, square root, and absolute value. Trigonometric functions and logarithmic/exponential functions are also included in this section.	4-30
4	Relational Functions	Describes how to use relational functions to compare two numbers for equality, non-equality, greater than, greater than or equal to, less than, and less than or equal to.	4-46
5	Bit Operation Functions	Describes how to perform comparison and move operations on bit strings.	4-53
6	Data Move Functions	Describes basic data move capabilities.	4-75
7	Data Table Functions	Describes how to use data table functions to enter values into and copy values out of a table.	4-124
8	Conversion Functions	Describes how to convert a data item from one number type to another.	4-150
9	Control Functions	Describes how to limit program execution and change the way the CPU executes the application program by using the control functions.	4-165

Section 1: Relay Functions

This section explains the use of contacts, coils, and links in ladder logic rungs.

Using Contacts

A contact is used to monitor the state of a reference. Whether the contact passes power flow depends on the state or status of the reference being monitored and on the contact type. A reference is ON if its state is 1; it is OFF if its state is 0.

Table 4-1. Types of Contacts

Type of Contact	Display	Contact Passes Power to Right:
Normally-open contact	–| |–	When reference is ON.
Normally-closed contact	–|/|–	When reference is OFF.
Positive transition contact	–|↑|–	If reference goes ON.
Negative transition contact	–|↓|–	If reference goes OFF.
Fault contact	–FAULT]–	If reference has point fault.
No fault contact	–NOFLT]–	If reference has no point fault.
High alarm contact	–HIALR]–	If reference exceeds high alarm.
Low alarm contact	–LOALR]–	If reference exceeds low alarm.
Continuation contact	<+>–––	If the preceding continuation coil is set ON.

Using Coils

Coils are used to control discrete references. Conditional logic must be used to control the flow of power to a coil. Coils cause action directly; they do not pass power flow to the right. If additional logic in the program should be executed as a result of the coil condition, an internal reference should be used for that coil or a continuation coil/contact combination may be used.

Coils are always located at the rightmost position of a line of logic. A rung may contain up to eight coils.

The type of coil used will depend on the type of program action desired. The states of retentive coils are saved when power is cycled or when the PLC goes from **STOP** to **RUN** mode. The states of non-retentive coils are set to zero when power is cycled or the PLC goes from **STOP** to **RUN** mode.

Table 4-2. Types of Coils

Type of Coil	Display	Power to Coil	Result
Coil (normally open)	–( )–	ON	Set reference ON.
		OFF	Set reference OFF.
Negated	–(/)–	ON	Set reference OFF.
		OFF	Set reference ON.
Retentive	–(M)–	ON	Set reference ON, retentive.
		OFF	Set reference OFF, retentive.
Negated Retentive	–(/M)–	ON	Set reference OFF, retentive.
		OFF	Set reference ON, retentive.
Positive Transition	–(↑)–	OFF→ON	If reference is OFF, set it ON for one sweep.
Negative Transition	–(↓)–	ON→OFF	If reference is OFF, set it ON for one sweep.
SET	–(S)–	ON	Set reference ON until reset OFF by -(R)-.
		OFF	Do not change the coil state.
RESET	–(R)–	ON	Set reference OFF until set ON by -(S)-.
		OFF	Do not change the coil state.
Retentive SET	–(SM)–	ON	Set reference ON until reset OFF by -(RM)-, retentive.
		OFF	Do not change the coil state.
Retentive RESET	–(RM)–	ON	Set reference OFF until set ON by -(SM)-, retentive.
		OFF	Do not change the coil state.
Continuation coil	– – – <+>	ON	Set next continuation contact ON.
		OFF	Set next continuation contact OFF.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Normally Open Contact —| |—

A normally open contact acts as a switch that passes power flow if the associated reference is ON (1).

Normally Closed Contact —|/|—

A normally closed contact acts as a switch that passes power flow if the associated reference is OFF (0).

Example:

The following example shows a rung with 10 elements having nicknames from E1 to E10. Coil E10 is ON when reference E1, E2, E5, E6, and E9 are ON and references E3, E4, E7, and E8 are OFF.

```
|
|  E1        E2        E3        E4        E5        E6        E7        E8        E9        E10
|--| |-------| |-------|/|-------|/|-------| |-------| |-------|/|-------|/|-------| |-------( )-
|
```

Positive Transition Contact —|↑|—

A positive transition contact passes power flow if the associated reference transitions from OFF (0) to ON (1). The transition is determined from the write of an OFF (0) to the next write of an ON (1) value. These writes may occur multiple times during a PLC sweep, resulting in the transition bit being set for only a portion of the sweep; or they may occur several PLC sweeps apart, resulting in the transition bit being set for more than one sweep.

Caution

Do not use the positive transition contact for references used with transition coils (also called one-shots) or SET and RESET coils.

Negative Transition Contact —|↓|—

A negative transition contact passes power flow if the associated reference transitions from ON (1) to OFF (0). The transition is determined from the write of an ON (1) to the next write of an OFF (0) value. These writes may occur multiple times during a PLC sweep, resulting in the transition bit being set for only a portion of the sweep; or they may occur several PLC sweeps apart, resulting in the transition bit being set for more than one sweep.

Caution

Do not use the negative transition contact for references used with transition coils (also called one-shots) or SET and RESET coils.

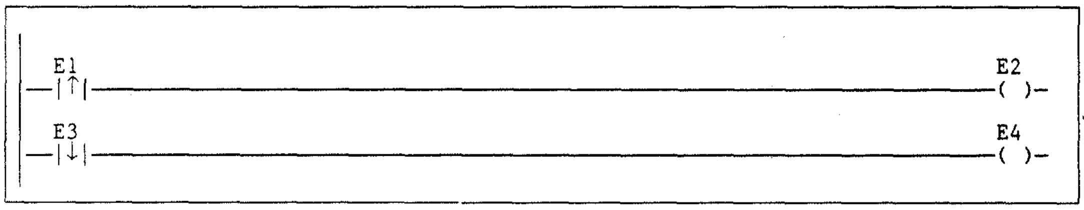
Additional Information on Transition Contacts

The transition bit for a reference point is affected when that point is written to. It is set when the point transitions from OFF to ON or from ON to OFF. It is reset when the state after the write is the same as the state before the write, i.e., ON to ON or OFF to OFF. The source of the write is immaterial; it can be an output coil, a function block output, the input scan, an input interrupt, a PCM SYSWRITE, a data change from the program, or Genius Datagram. When the point is written, the transition bit is immediately affected. Transition bits are not changed by the scan itself; only a write to the reference point. A write must be made to a reference in order to clear the transition bit, or it will appear to be "stuck." Nothing is done automatically per sweep to clear transition bits, except for configured input points, where a transition is cleared when the input data is read and the input point is in the same state as when read the previous sweep.

Overrides do not protect transition bits. If a write occurs to an overridden point, the transition bit is cleared. For example, the transition bit of an overridden input point is cleared when the input is scanned.

Example 1:

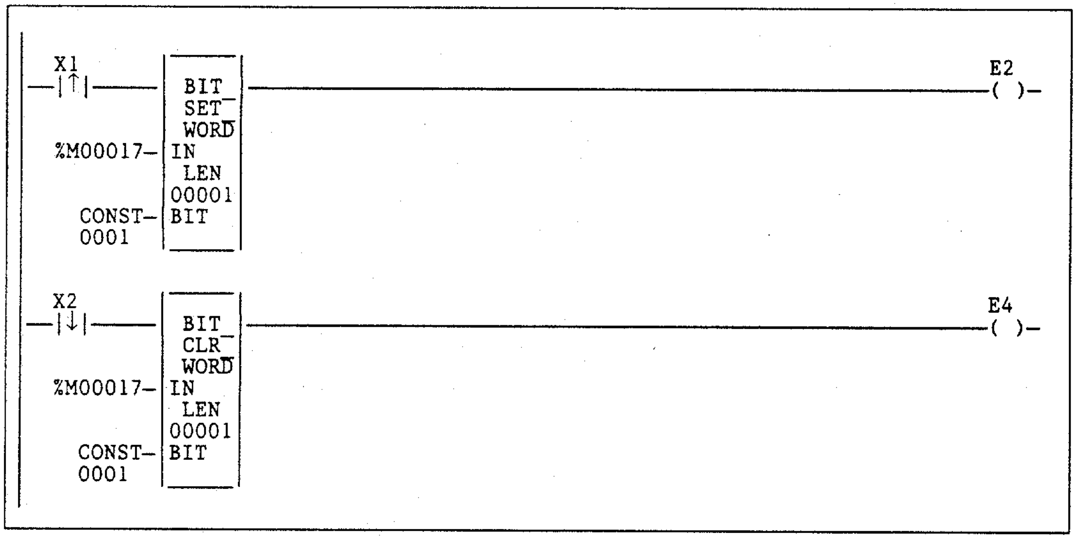
The following example shows the use of positive and negative transition contacts. Coil E2 is on for one logic sweep when element E1 transitions from OFF to ON. Coil E4 is ON for one sweep when element E3 transitions from ON to OFF.



Example 2:

In the following example, bit %M00017 is set by a BSET function and then cleared by a BCLR function. The positive transition contact X1 activates the BSET, and the negative transition X2 activates the BCLR.

The positive transition associated with bit %M00017 will be on until %M00017 is reset by the BCLR function. This occurs because the bit is only written when contact X1 goes from OFF to ON. Similarly, the negative transition associated with bit %M00017 will be ON until %M00017 is set by the BSET function.



Fault Contact -[FAULT]-

Fault contacts are used to detect faults in discrete or analog machine references, or to locate faults (rack, slot, bus). The use of fault contacts must be enabled during CPU configuration if point fault reporting is desired. (Refer to chapter 2, section 3, "Program Organization and User Data," for more information on point faults.)

A fault contact passes power flow if its associated variable has a point fault.

Note

Use the machine reference address (%I, %Q, %AI, %AQ) with the FAULT/NOFLT contacts to guarantee correct indication of module status. (When using the rack, slot, bus, module reference with a FAULT/NOFLT contact, the fault indication of a given module is cleared when the associated fault is cleared from the fault table.

No Fault Contact -[NOFLT]-

No fault contacts are also used to detect faults in discrete or analog machine references. Their use must also be enabled if point fault reporting is desired. (Refer to chapter 2, section 3, "Program Organization and User Data," for more information on point faults.)

A no fault contact passes power flow if its associated discrete variable does not have a point fault.

Note

Use the machine reference address (%I, %Q, %AI, %AQ) with the FAULT/NOFLT contacts to guarantee correct indication of module status. (When using the rack, slot, bus, module reference with a FAULT/NOFLT contact, the fault indication of a given module is cleared when the associated fault is cleared from the fault table.

High Alarm Contact -[HIALR]-

The high alarm contact is used to detect a high alarm associated with an analog reference. Use of this contact and the low alarm contact must be enabled during CPU configuration. (From the configuration software menu, select "Memory Allocation and Point Fault Enable". Change the setting for point fault reference from **DISABLED** to **ENABLED**).

A high alarm contact passes power flow if the high alarm bit associated with the analog reference is ON.

Low Alarm Contact -[LOALR]-

The low alarm contact is used to detect a low alarm associated with an analog reference. Use of this contact must be enabled during CPU configuration, as described above for the high alarm contact.

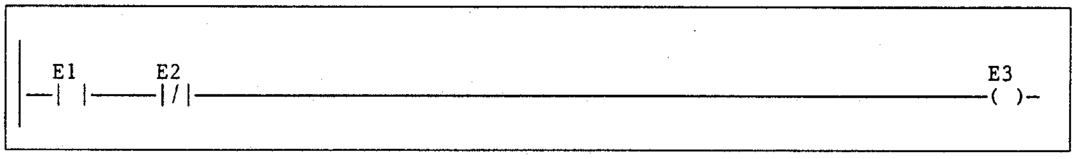
A low alarm contact passes power flow if the low alarm bit associated with the analog reference is ON.

Coil -( )-

A coil sets a discrete reference ON while it receives power flow. It is non-retentive; therefore, it cannot be used with system status references (%SA, %SB, %SC, or %G).

Example:

In the following example, coil E3 is ON when reference E1 is ON and reference E2 is OFF.

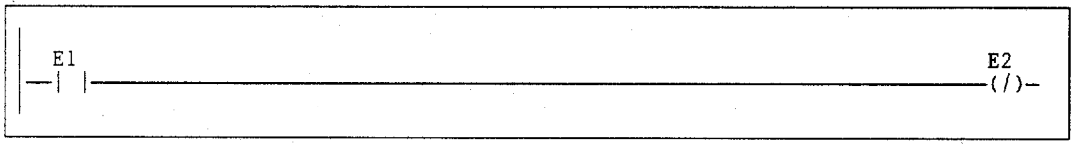


Negated Coil -(/)-

A negated coil sets a discrete reference ON when it does not receive power flow. It is not retentive; therefore, it cannot be used with system status references (%SA, %SB, %SC, or %G).

Example:

In the following example, negated coil E2 is ON when reference E1 is OFF.



Retentive Coil -(M)-

Like a normally open coil, the retentive coil sets a discrete reference ON while it receives power flow. The state of the retentive coil is retained across power failure. Therefore, it cannot be used with references from strictly non-retentive memory (%T).

Negated Retentive Coil -(/M)-

The negated retentive coil sets a discrete reference ON when it does not receive power flow. The state of the negated retentive coil is retained across power failure. Therefore, it cannot be used with references from strictly non-retentive memory (%T).

Positive Transition Coil —(↑)—

If the reference associated with a positive transition coil is OFF, when the coil receives power flow it is set to ON until the next time the coil is executed. (If the rung containing the coil is skipped on subsequent sweeps, it will remain ON.) This coil can be used as a one-shot.

Do not write from external devices (e.g., PCM) to references used on positive transition coils since it will destroy the one-shot nature of these coils.

Note

Do not use transition contacts on positive transition coils since the coil uses the transition bit to store the power flow value into the coil.

Transitional coils can be used with references from either retentive or non-retentive memory (%Q, %M, %T, %G, %SA, %SB, or %SC). ↓

Negative Transition Coil —(↓)—

If the reference associated with this coil is OFF, when the coil ***stops*** receiving power flow, the reference is set to ON until the next time the coil is executed.

Do not write from external devices to references used on negative transition coils since it will destroy the one-shot nature of these coils.

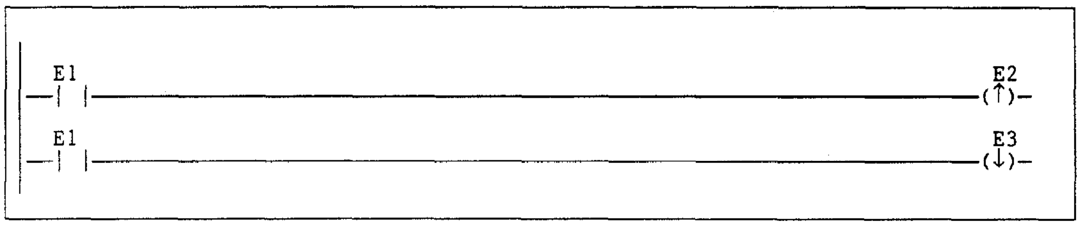
Note

Do not use transition contacts on negative transition coils since the coil uses the transition bit to store the power flow value into the coil.

Transitional coils can be used with references from either retentive or non-retentive memory (%Q, %M, %T, %G, %SA, %SB, or %SC).

Example:

In the following example, when reference E1 goes from OFF to ON, coils E2 and E3 receive power flow, turning E2 ON for one logic sweep. When E1 goes from ON to OFF, power flow is removed from E2 and E3, turning coil E3 ON for one sweep.



SET Coil -(S)-

SET and RESET are non-retentive coils that can be used to keep ("latch") the state of a reference (e.g., E1) either ON or OFF. When a SET coil receives power flow, its reference stays ON (whether or not the coil itself receives power flow) until it is reset by power flow to a RESET coil of the same reference (e.g., E1 in the example below).

Note

SET coils write an undefined result to the transition bit for the given reference. *Do not use transition contacts on references used on SET coils.*

RESET Coil -(R)-

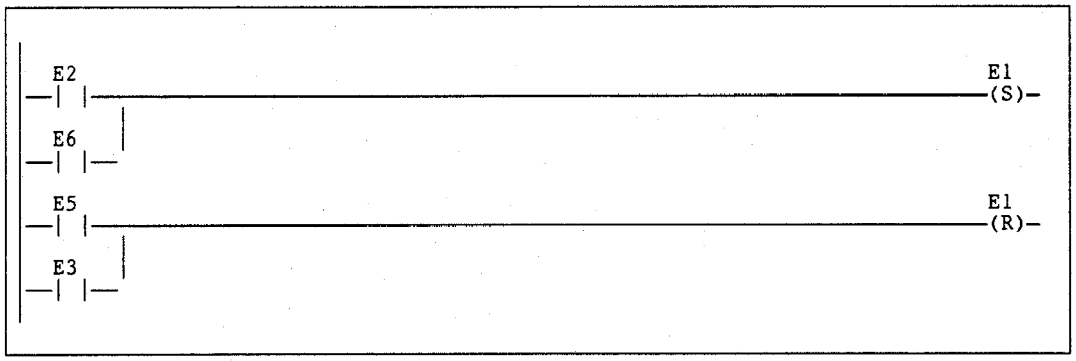
The RESET coil sets a discrete reference OFF if the coil receives power flow. The reference remains OFF until set ON by a SET coil of the same reference (e.g., E1 in the example below). The last-solved SET coil or RESET coil of a pair takes precedence.

Note

RESET coils write an undefined result to the transition bit for the given reference. *Do not use transition contacts on references used on RESET coils.*

Example:

In the following example, the coil represented by E1 is turned ON whenever reference E2 or E6 is ON. The coil represented by E1 is turned OFF whenever reference E5 or E3 is ON.



Retentive SET Coil –(SM)–

Retentive SET and RESET coils are similar to SET and RESET coils, but they are retained across power failure or when the PLC transitions from **STOP** to **RUN** mode. A retentive SET coil sets a discrete reference ON if the coil receives power flow. The reference remains ON until reset by a retentive RESET coil.

Note

Retentive SET coils write an undefined result to the transition bit for the given reference. Do not use transition contacts on references used on retentive SET coils. (Refer to the information on "Transitions and Overrides" in chapter 2.)

Retentive RESET Coil –(RM)–

This coil sets a discrete reference OFF if it receives power flow. The reference remains OFF until set by a retentive SET coil. The state of this coil is retained across power failure or when the PLC transitions from **STOP** to **RUN** mode.

Note

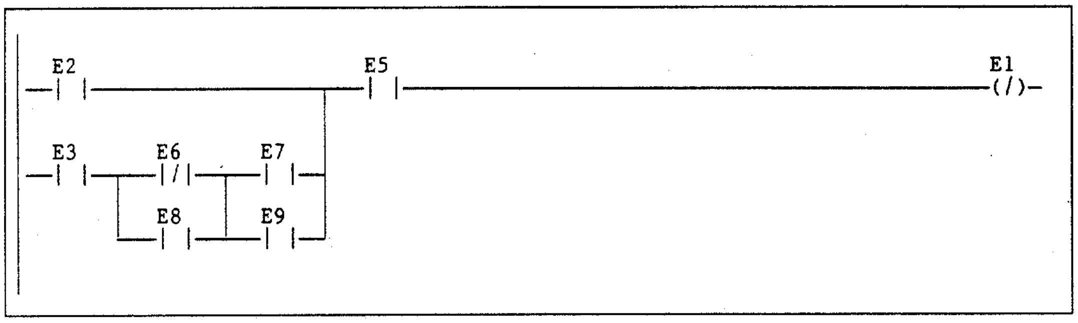
Retentive RESET coils write an undefined result to the transition bit for the given reference. Do not use transition contacts on references used on retentive RESET coils. (Refer to the information on "Transitions and Overrides" in chapter 2.)

Links

Horizontal and vertical links are used to connect elements of a line of ladder logic between functions. Their purpose is to complete the flow of logic ("power") from left to right in a line of logic.

Example:

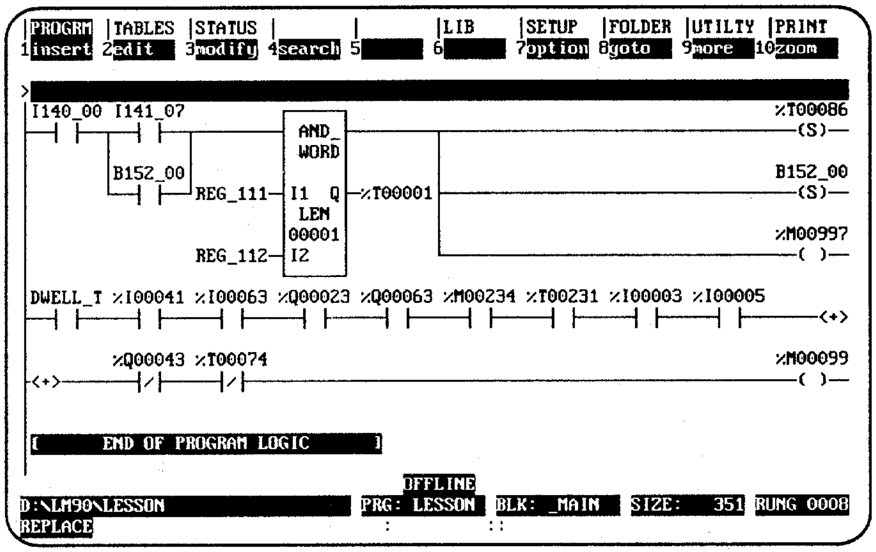
In the following example, two horizontal links are used to connect contacts E2 and E5. A vertical link is used to connect contacts E3, E6, E7, E8, and E9 to E2.



Continuation Coils (– – –<+>) and Contacts (<+>– – –)

Continuation coils (– – –<+>) and contacts (<+>– – –) are used to continue relay ladder logic beyond the limit of ten columns. The state of the last executed continuation coil is the flow state that will be used on the next executed continuation contact. If the flow of logic does not execute a continuation coil before it executes a continuation contact, the state of the contact will be no flow.

There can be only one continuation coil and contact per rung; the continuation contact must be in column 1 and the continuation coil must be in column 10. An example continuation coil and contact are shown below.



Section 2: Timers and Counters

This section explains how to use on-delay, off-delay, and stopwatch-type timers, up counters, and down counters. The data associated with these functions is retentive through power cycles.

Abbreviation	Function	Page
ONDTR	Retentive On-Delay Timer	4-15
OFDT	Off-Delay Timer	4-18
TMR	Simple On-Delay Timer	4-21
UPCTR	Up Counter	4-24
DNCTR	Down Counter	4-26

Function Block Data Required for Timers and Counters

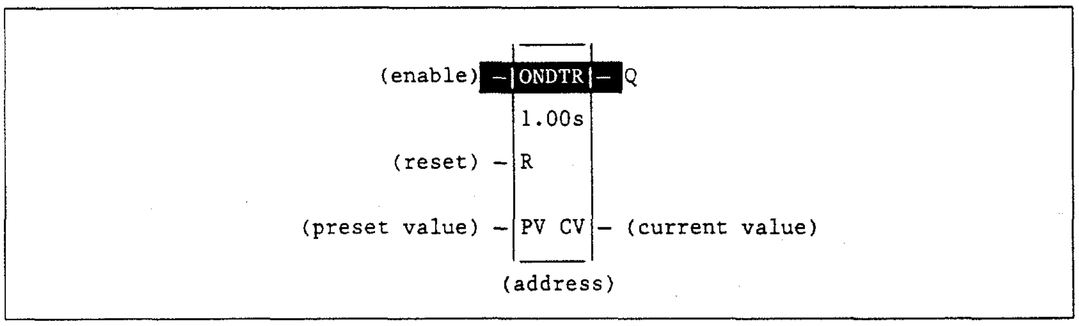
Each timer or counter uses three words (registers) of %R, %P, or %L memory to store the following information:

current value (CV)	word 1
preset value (PV)	word 2
control word	word 3

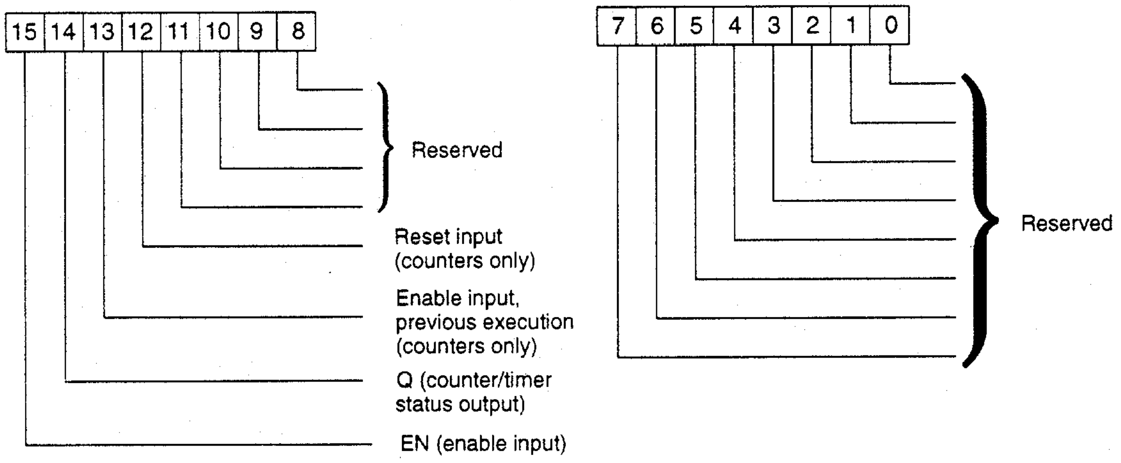
Note

Do not allow other functions to write to these registers when using the timer function.

When you enter a timer or counter, you must enter an address for the location of these three words (registers) directly below the graphic representing the function. For example:



The control word stores the state of the boolean input and output of its associated function block, as shown in the following format:


15 14 13 12 11 10 9 8
7 6 5 4 3 2 1 0
Reserved
Reset input (counters only)
Enable input, previous execution (counters only)
Q (counter/timer status output)
EN (enable input)
Reserved

Bits 0 through 13 are used for timer accuracy; bits 0 through 10 are not used for counters.

Note

Do not write to these from some other function when using the timer function.

ONDTR

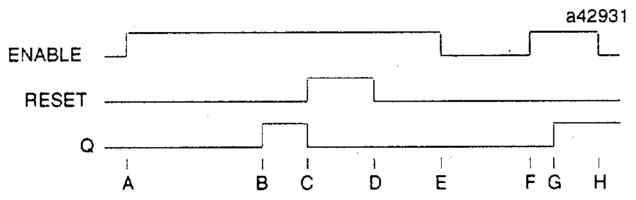
A retentive on-delay timer (ONDTR) increments while it receives power flow and holds its value when power flow stops. Time may be counted in seconds (the default selection), tenths of seconds, or hundredths of seconds. The range is 0 to +32,767 time units. The state of this timer is retentive on power failure; no automatic initialization occurs at power-up.

When the ONDTR first receives power flow, it starts accumulating time (current value). When this timer is encountered in the ladder logic, its current value is updated.

Note

> If multiple occurrences of the same timer with the same reference address are enabled during a CPU sweep, the current values of the times will be the same.

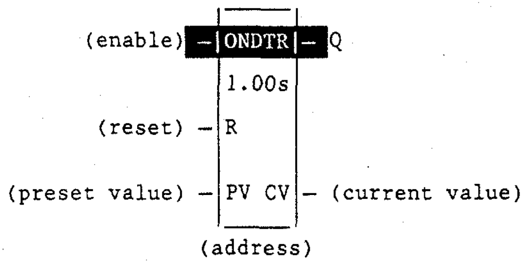
When the current value (CV) equals or exceeds the preset value (PV), output Q is energized. As long as the timer continues to receive power flow, it continues accumulating until the maximum value is reached. Once the maximum value is reached, it is retained and output Q remains energized regardless of the state of the enable input.



```
A  =   ENABLE goes high; timer starts accumulating.
B  =   CV reaches PV; Q goes high.
C  =   RESET goes high; Q goes low, accumulated time is reset.
D  =   RESET goes low; timer then starts accumulating again.
E  =   ENABLE goes low; timer stops accumulating.
       Accumulated time stays the same.
F  =   ENABLE goes high again; timer continues accumulating time.
G  =   CV becomes equal to PV; Q goes high.
       Timer continues to accumulate
       time until ENABLE goes low,RESET goes high,
       or current value becomes equal to the maximum time.
H  =   ENABLE goes low; timer stops accumulating time.
```

When power flow to the timer stops, the current value stops incrementing and is retained. Output Q, if energized, will remain energized. When the function receives power flow again, the current value again increments, beginning at the retained value. When reset (R) receives power flow, the current value is set back to zero and output (Q) is de-energized unless PV equals zero.

If PV equals zero and the timer is enabled, the output of the timer will activate. Subsequent removal of enable or activation of reset will have no effect on the timer output; it will remain enabled.



When the ONDTR is used in a program block that is not called every sweep, the timer accumulates time between calls to the program block unless it is reset. This means that it functions like a timer operating in a program with a much slower sweep than the timer in the main program block. For program blocks that are inactive for a long time, the timer should be programmed to allow for this catch-up feature.

For example, if a timer in a program block is reset and the program block is not called (is inactive) for 4 minutes, when the program block is called, four minutes of time will already have accumulated. This time is applied to the timer when enabled, unless the timer is first reset.

Parameters:

Parameter	Description
address	The ONDTR uses three consecutive words (registers) of %R, %P, or %L memory to store the: • Current value (CV) = word 1. • Preset value (PV) = word 2. • Control word = word 3. When you enter an ONDTR, you must enter an address for the location of these three consecutive words (registers) directly below the graphic representing the function. For more information, refer to page 4-13. Note: Do not use this address with other instructions. Caution: Overlapping references will result in erratic operation of the timer.
enable	When enable receives power flow, the timer's current value is incremented.
R	When R receives power flow, it resets the current value to zero.
PV	PV is the value to copy into the timer's preset value when the timer is enabled or reset. For a register (%R) PV reference, the PV parameter is specified as the second word of the address parameter. For example, an address parameter of %R00001 would use %R00002 as the PV parameter.
Q	Output Q is energized when the current value is greater than or equal to the preset value. Since no automatic initialization to the Q state occurs at power-up, the Q state is retentive across power failure.
CV	CV contains the current value of the timer.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
address									•	•	•			•		
enable	•															
R	•															
PV	•	•	•	•	•		•		•	•	•	•	•	•	•	•
Q	•															•
CV	•	•	•	•	•		•		•	•	•	•	•	•		•

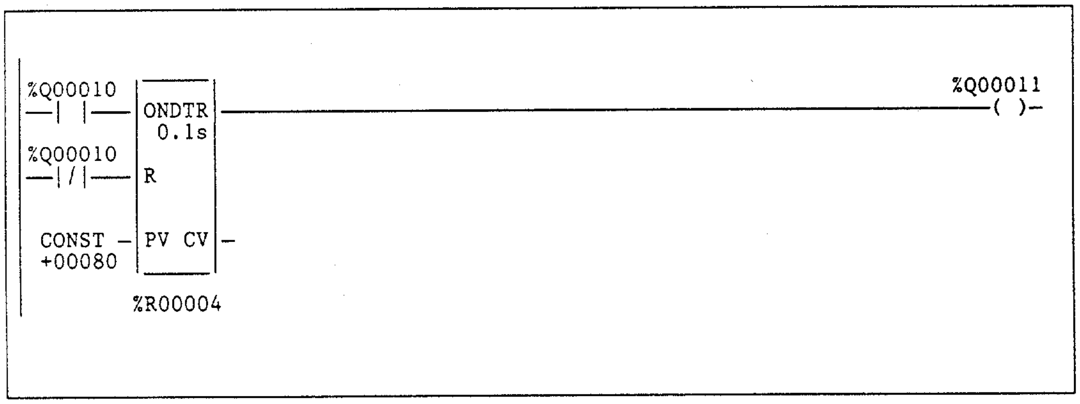
• Valid reference or place where power may flow through the function.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, a retentive on-delay timer is used to create a signal (%Q00011) that turns on 8.0 seconds after %Q00010 turns on, and turns off when %Q00010 turns off.



OFDT

The off-delay timer (OFDT) increments while power flow is off, and resets to zero when power flow is on. Time may be counted in seconds (the default selection), tenths of seconds, or hundredths of seconds. The range is 0 to +32767 time units. The state of this timer is retentive on power failure; no automatic initialization occurs at power-up.

When the OFDT first receives power flow, it passes power to the right and the current value (CV) is set to zero. The output remains on as long as the function receives power flow. If the function stops receiving power flow from the left, it continues to pass power to the right and the timer starts accumulating time in CV.

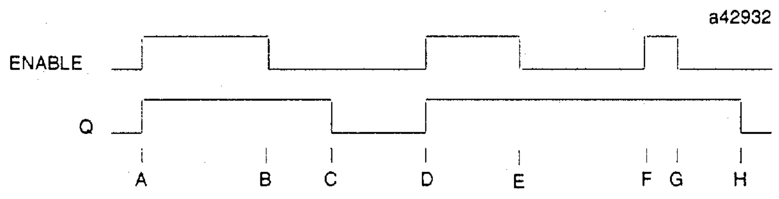
Note

If multiple occurrences of the same timer with the same reference address are enabled during a CPU sweep, the current values of the times will be the same.

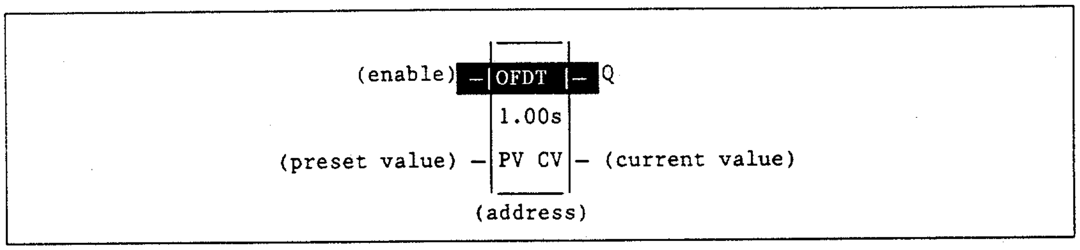
The OFDT does not pass power flow if the preset value is zero or negative.

Each time the function is invoked with the enabling logic set OFF, the current value is updated to reflect the elapsed time since the timer was turned off. When the current value (CV) is equal to the preset value (PV), the function stops passing power flow to the right. The current value never exceeds the preset value.

When the function receives power flow again, the current value resets to zero.



```
A  =   ENABLE and Q both go high.
B  =   ENABLE goes low; timer starts accumulating time.
C  =   CV reaches PV; Q goes low, and timer stops
       accumulating time.
D  =   ENABLE goes high; timer is reset (CV = 0).
E  =   ENABLE goes low; timer starts accumulating time.
F  =   ENABLE goes high; timer is reset (CV = 0).
G  =   ENABLE goes low; timer begins accumulating time.
H  =   CV reaches PV; Q goes low, and timer stops
       accumulating time.
```



When the OFDT is used in a program block that is not called every sweep, the timer accumulates time between calls to the program block unless it is reset. This means that it functions like a timer operating in a program with a much slower sweep than the timer in the main program block. For program blocks that are inactive for a long time, the timer should be programmed to allow for this catch-up feature.

For example, if a timer in a program block is reset and the program block is not called (is inactive) for 4 minutes, when the program block is called, four minutes of time will already have accumulated. This time is applied to the timer when enabled, unless the timer is first reset.

Parameters:

Parameter	Description
address	The OFDT uses three consecutive words (registers) of %R, %P, or %L memory to store the: • Current value (CV) = word 1. • Preset value (PV) = word 2. • Control word = word 3. When you enter an OFDT, you must enter an address for the location of these three consecutive words (registers) directly below the graphic representing the function. For more information, refer to page 4-13. **Note:** Do not use this address with other instructions. **Caution:** Overlapping references will result in erratic operation of the timer.
enable	When enable receives power flow, the timer's current value is incremented.
PV	PV is the value to copy into the timer's preset value when the timer is enabled or reset. For a register (%R) PV reference, the PV parameter is specified as the second word of the address parameter. For example, an address parameter of %R00001 would use %R00002 as the PV parameter.
Q	Output Q is energized when the current value is less than the preset value. Since no automatic initialization to the Q state occurs at power-up, the Q state is retentive across power failure.
CV	CV contains the current value of the timer.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
address									•	•	•			•		
enable	•															
PV	•	•	•	•	•		•		•	•	•	•	•	•	•	•
Q	•															•
CV	•	•	•	•	•		•		•	•	•	•	•	•		•

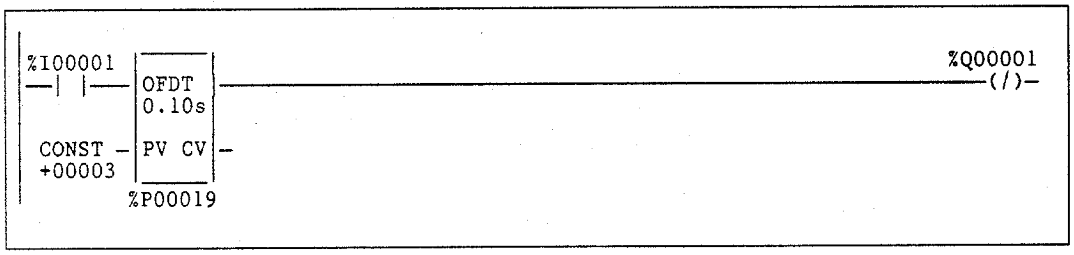
• Valid reference or place where power may flow through the function.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, an OFDT timer is used to turn off an output (%Q00001) whenever an input (%I00001) turns on. The output is turned on again 0.3 seconds after the input goes off.



TMR

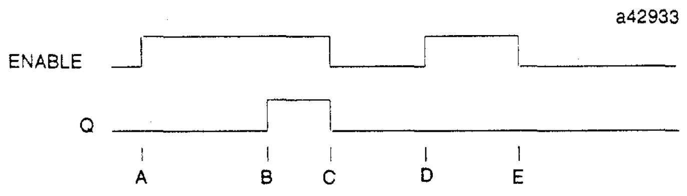
The simple on-delay timer (TMR) function increments while it receives power flow and resets to zero when power flow stops. Time may be counted in seconds (the default selection), tenths of seconds, or hundredths of seconds. The range is 0 to +32,767 time units. The state of this timer is retentive on power failure; no automatic initialization occurs at power-up.

When the TMR receives power flow, the timer starts accumulating time (current value). The current value is updated when it is encountered in the logic to reflect the total elapsed time since the timer was last reset.

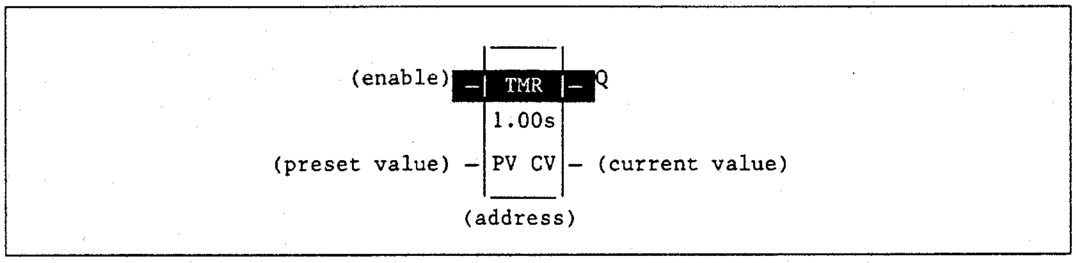
Note

If multiple occurrences of the same timer with the same reference address are enabled during a CPU sweep, the current values of the times will be the same. Additionally, a TMR timer will expire (pass power flow to the right) the first sweep that it is enabled if the previous sweep time was greater than the preset value (PV).

This update occurs as long as the enabling logic remains ON. When the current value (CV) equals or exceeds the preset value (PV), the function begins passing power flow to the right. The timer continues accumulating time until the maximum value is reached. When the enabling parameter transitions from ON to OFF, the timer stops accumulating time and the current value is reset to zero.



```
A  =   ENABLE goes high; timer begins accumulating time.
B  =   CV reaches PV; Q goes high, and
       timer continues accumulating time.
C  =   ENABLE goes low; Q goes low; timer stops accumulating
       time and CV is cleared.
D  =   ENABLE goes high; timer starts accumulating time.
E  =   ENABLE goes low before CV reaches PV; Q remains low;
       timer stops accumulating time and is cleared to zero.
```



When the TMR is used in a program block that is not called every sweep, the timer accumulates time between calls to the program block unless it is reset. This means that it functions like a timer operating in a program with a much slower sweep than the timer in the main program block. For program blocks that are inactive for a long time, the timer should be programmed to allow for this catch-up feature.

For example, if a timer in a program block is reset and the program block is not called (is inactive) for 4 minutes, when the program block is called, four minutes of time will already have accumulated. This time is applied to the timer when enabled, unless the timer is first reset.

Parameters:

Parameter	Description
address	The TMR uses three consecutive words (registers) of %R, %P, or %L memory to store the: • Current value (CV) = word 1. • Preset value (PV) = word 2. • Control word = word 3. When you enter an TMR, you must enter an address for the location of these three consecutive words (registers) directly below the graphic representing the function. For more information, refer to page 4-13. Note: Do not use this address with other instructions. Caution: Overlapping references will result in erratic operation of the timer.
enable	When enable receives power flow, the timer's current value is incremented. When the TMR is not enabled, the current value is reset to zero and Q is turned off.
PV	PV is the value to copy into the timer's preset value when the timer is enabled or reset. For a register (%R) PV reference, the PV parameter is specified as the second word of the address parameter. For example, an address parameter of %R00001 would use %R00002 as the PV parameter.
Q	Output Q is energized when TMR is enabled and the current value is greater than or equal to the preset value.
CV	CV contains the current value of the timer.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
address									•	•	•			•		
enable	•															
PV	•	•	•	•	•		•		•	•	•	•	•	•	•	•
Q	•															•
CV	•	•	•	•	•		•		•	•	•	•	•	•		•

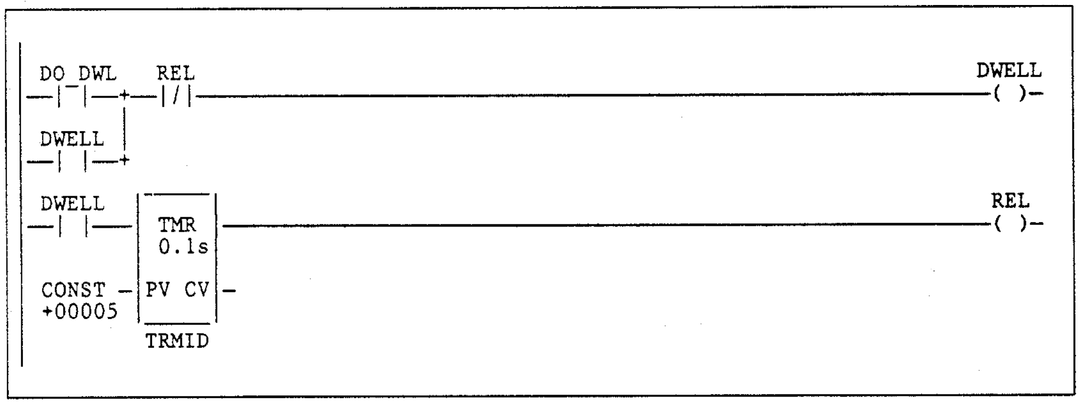
• Valid reference or place where power may flow through the function.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

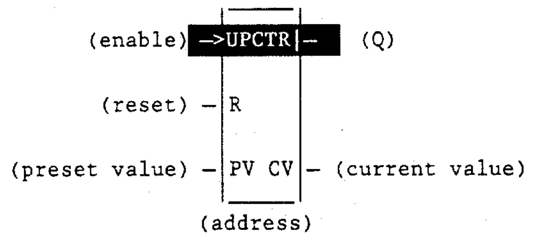
In the following example, an on-delay timer TMRID is used to control the length of time that coil DWELL is on. When the normally open (momentary) contact DO_DWL is on, coil DWELL is energized. The contact of coil DWELL keeps coil DWELL energized (when contact DO_DWL is released), and also starts the timer TMRID. When TMRID reaches its preset value of one-half second, coil REL energizes, interrupting the latched-on condition of coil DWELL. The contact DWELL interrupts power flow to TMRID, resetting its current value and de-energizing coil REL. The circuit is then ready for another momentary activation of contact DO_DWL.



UPCTR

The Up Counter (UPCTR) function is used to count up to a designated value. The range is 0 to +32,767 counts. When the up counter reset is ON, the current value (CV) of the counter is reset to 0. Each time the enable input transitions from OFF to ON, the current value is incremented by 1. The current value (CV) can be incremented past the preset value (PV). The output is ON whenever the current value is greater than or equal to the preset value.

The output state (Q) of the UPCTR is retentive on power failure; no automatic initialization occurs at power-up.



Parameters:

Parameter	Description
address	The UPCTR uses three consecutive words (registers) of %R, %P, or %L memory to store the: • Current value (CV) = word 1. • Preset value (PV) = word 2. • Control word = word 3. When you enter an UPCTR, you must enter an address for the location of these three consecutive words (registers) directly below the graphic representing the function. For more information, refer to page 4-13. <u>Note:</u> Do not use this address with another up counter, down counter, or any other instruction or improper operation will result. <u>**Caution:**</u> Overlapping references will result in erratic operation of the counter.
enable	On a positive transition of enable, the current count is incremented by one.
R	When R receives power flow, it resets the current value back to zero.
PV	PV is the value to copy into the counter's preset value when the counter is enabled or reset. For a register (%R) PV reference, the PV parameter is specified as the second word of the address parameter. For example, an address parameter of %R00001 would use %R00002 as the PV parameter.
Q	Output Q is energized when the current value is greater than or equal to the preset value. The Q state is retentive on power failure; no automatic initialization occurs at power-up.
CV	CV contains the current value of the counter.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
address									•	•	•			•		
enable	•															
R	•															
PV	•	•	•	•	•		•		•	•	•	•	•	•	•	•
Q	•															•
CV	•	•	•	•	•		•		•	•	•	•	•	•		•

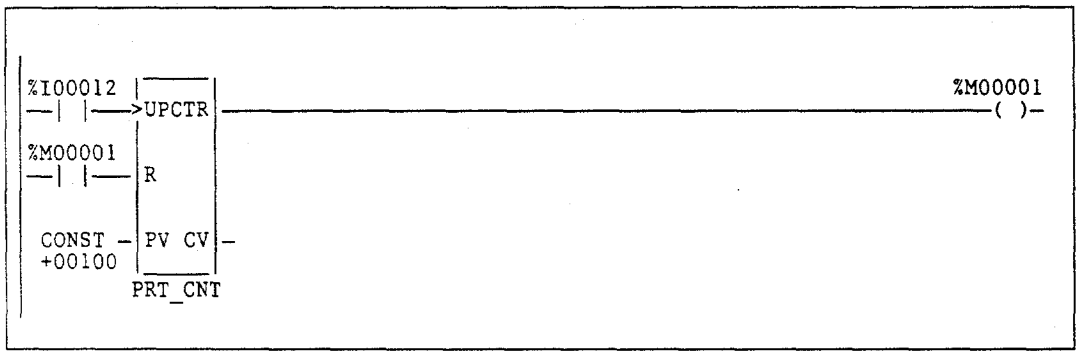
• Valid reference or place where power may flow through the function.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

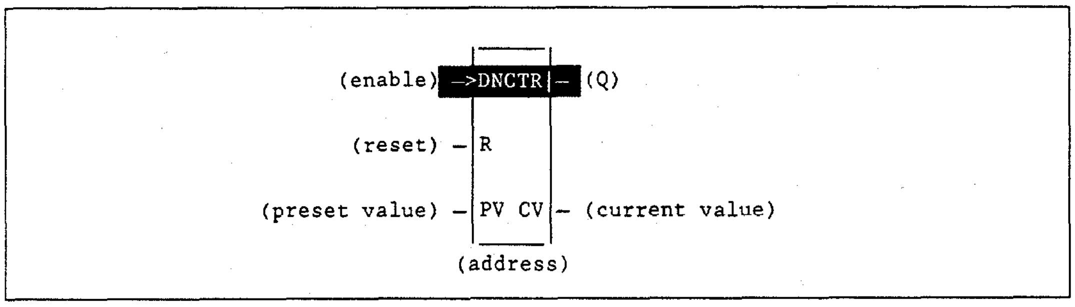
Example:

In the following example, every time input %I00012 transitions from OFF to ON, up counter PRT_CNT counts up by 1; internal coil %M00001 is energized whenever 100 parts have been counted. Whenever %M00001 is ON, the accumulated count is reset to zero.



DNCTR

The Down Counter (DNCTR) function is used to count down to zero from a preset value. The minimum preset value is zero; the maximum preset value is +32,767 counts. The minimum current value is −32,768. When reset, the current value (CV) of the counter is set to the preset value (PV). When the enable input transitions from OFF to ON, the current value is decremented by one. The output is ON whenever the current value is less than or equal to zero.



Parameters:

Parameter	Description
address	The DNCTR uses three consecutive words (registers) of %R, %P, or %L memory to store the: • Current value (CV) = word 1. • Preset value (PV) = word 2. • Control word = word 3. When you enter an DNCTR, you must enter an address for the location of these three consecutive words (registers) directly below the graphic representing the function. For more information, refer to page 4-13. **Note:** Do not use this address with another down counter, up counter, or any other instruction or improper operation will result. **Caution:** Overlapping references will result in erratic operation of the counter.
enable	On a positive transition of enable, the current value is decremented by one.
R	When R receives power flow, it resets the current value to the preset value.
PV	PV is the value to copy into the counter's preset value when the counter is enabled or reset. For a register (%R) PV reference, the PV parameter is specified as the second word of the address parameter. For example, an address parameter of %R00001 would use %R00002 as the PV parameter.
Q	Output Q is energized when the current value is less than or equal to zero. The Q state is retentive on power failure; no automatic initialization occurs at power-up.
CV	CV contains the current value of the counter.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
address									•	•	•			•		
enable	•															
R	•															
PV	•	•	•	•	•		•		•	•	•	•	•	•	•	•
Q	•															•
CV	•	•	•	•	•		•		•	•	•	•	•	•		•

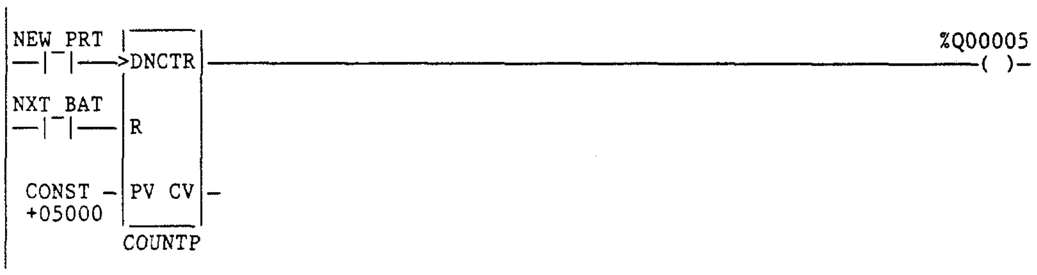
• Valid reference or place where power may flow through the function.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

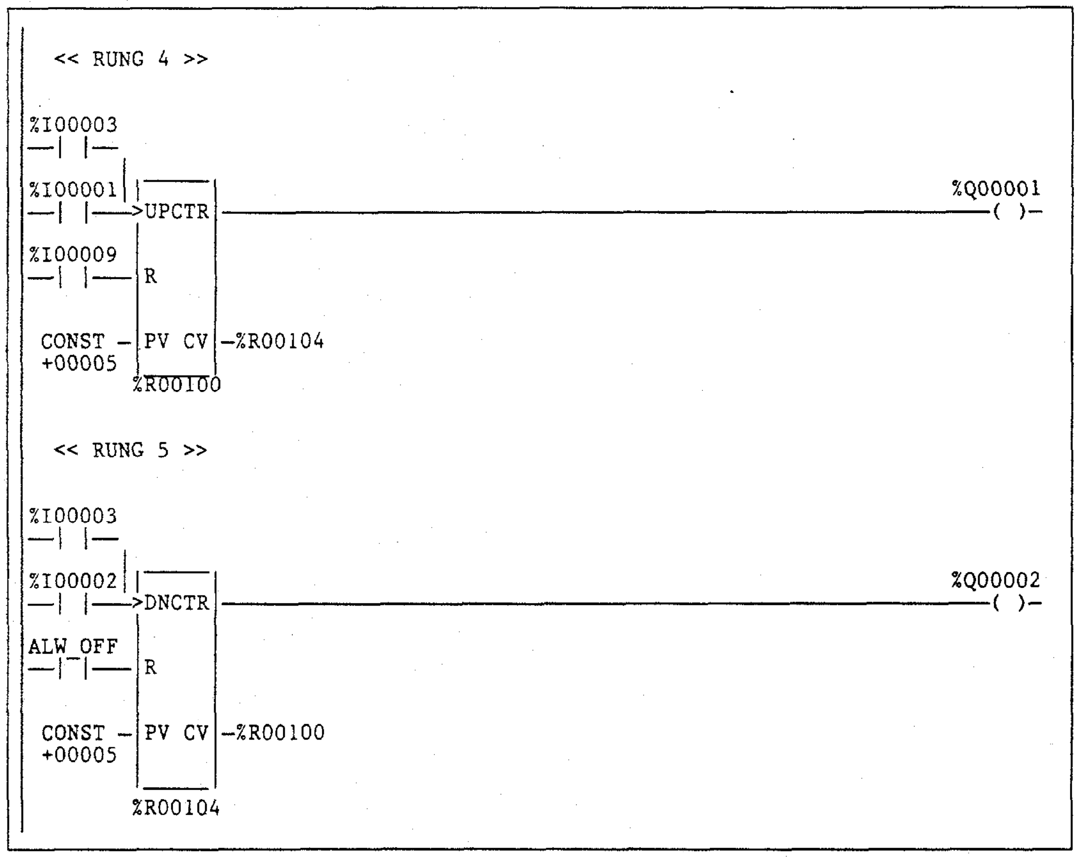
In the following example, the down counter identified as COUNTP counts 5000 new parts before energizing output %Q00005.



Up/Down Counter Pair Example:

In the following example, the PLC is used to keep track of the number of parts contained in a temporary storage area. There are two ways of accomplishing this function using the Series 90-70 instruction set.

The first method is to use an up/down counter pair with a shared register for the accumulated or current value. When the parts enter the storage area, the up counter increments by 1, increasing the current value of the parts in storage by a value of 1. When a part leaves the storage area, the down counter decrements by 1, decreasing the inventory storage value by 1. To avoid conflict with the shared register, both counters use different register addresses but each has a current value (CV) address that is the same as the accumulated value for the other register. This is the preferred method for this application.



The second method, shown below, uses the ADD and SUB functions to provide storage tracking.

```
|
|  << RUNG 7 >>
|
|%I00004  |‾‾‾‾‾‾‾|
|——|↑|——-|  ADD_ |-
|         |  INT‾ |
|         |       |
|%R00201—|I1   Q|—%R00201
|         |       |
|         |       |
| CONST  —|I2     |
| +00001  |_______|
|
|  << RUNG 8 >>
|
|%I00005  |‾‾‾‾‾‾‾|
|——|↑|——-|  SUB_ |-
|         |  INT‾ |
|         |       |
|%R00201—|I1   Q|—%R00201
|         |       |
|         |       |
| CONST  —|I2     |
| +00001  |_______|
|
```

Section 3: Math Functions

This section describes the Logicmaster 90 math functions:

Abbreviation	Function	Description	Page
ADD	Addition	Add two numbers.	4-31
SUB	Subtraction	Subtract one number from another.	4-31
MUL	Multiplication	Multiply two numbers.	4-31
DIV	Division	Divide one number by another, yielding a quotient.	4-31
MOD	Modulo Division	Divide one number by another, yielding a remainder.	4-34
SQRT	Square Root	Find the square root of an integer or real value.	4-36
ABS	Absolute Value	Find the absolute value of an integer, double precision integer, or real value.	4-38
SIN, COS, TAN, ASIN, ACOS, ATAN	Trigonometric Functions	Perform the appropriate function on the real value in input IN.	4-40
LOG, LN EXP, EXPT	Logarithmic/Exponential Functions	Perform the appropriate function on the real value in input IN.	4-42
RAD, DEG	Radian Conversion	Perform the appropriate function on the real value in input IN.	4-44

Note

Division and modulo division are similar functions which differ in their output; division finds a quotient, while modulo division finds a remainder.

MATH (ADD, SUB, MUL, DIV)

Math functions include addition, subtraction, multiplication, and division. When a function receives power flow, the appropriate math function is performed on input parameters I1 and I2. These parameters must be the same data type. Output Q is the same data type as I1 and I2.

Note

DIV rounds down; it does not round to the closest integer. (For example 24 DIV 5 = 4.)

Math functions operate on these types of data:

Data Type	Description
INT	Signed integer.
UINT	Unsigned integer.
DINT	Double precision signed integer.
REAL	Floating point.
MIXED	Mixed is available for MUL and DIV only.

Note

MUL_MIXED inputs are the same as INT; MUL_MIXED output is the same as DINT. DIV_MIXED input I1 is the same as DINT; DIV_MIXED input I2 and outputs are the same as INT.

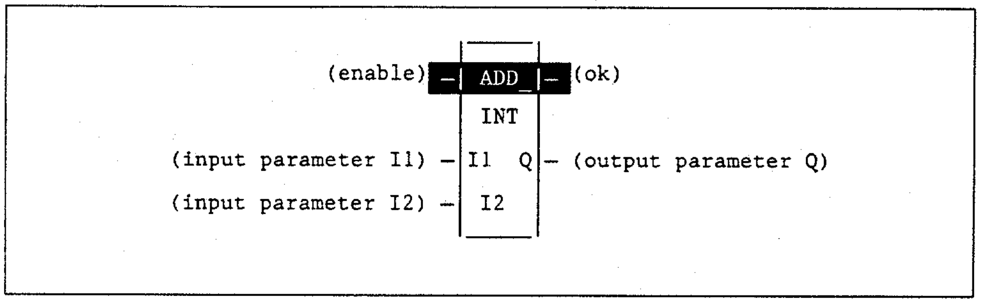
The default data type is signed integer; however, it can be changed after selecting the function. For more information on data types, refer to chapter 2, section 3, "Program Organization and User Data."

If the operation of INT or DINT operands results in overflow, the output reference is set to its largest possible value for the data type. For signed numbers, the sign is set to show the direction of the overflow. If signed or double precision integers are used, the sign of the result for DIV and MUL functions depends on the signs of I1 and I2. If the operation of UINT operands results in overflow, the output reference is set to the overflow value.

If the operation does not result in overflow, the ok output is set ON unless one of these invalid REAL operations occurs:

- For ADD, $(+\infty) + (-\infty)$.
- For SUB, $(\pm\infty) - (\pm\infty)$.
- For MUL, $0 \times (\infty)$.
- For DIV, 0 divided by 0.
- For DIV, ∞ divided by ∞.
- I1 and/or I2 is NaN (Not a Number).

In these cases, ok is set OFF.



Parameters:

Parameter	Description
enable	When the function is enabled, the operation is performed.
I1	I1 contains a constant or reference for the first value used in the operation. (I1 is on the left side of the mathematical equation, as in I1 – I2.)
I2	I2 contains a constant or reference for the second value used in the operation. (I2 is on the right side of the mathematical equation, as in I1 – I2.)
ok	The ok output is energized when the function is performed without dividing by zero, unless an invalid operation occurs or I1 and/or I2 is NaN.
Q	Output Q contains the result of the operation.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
I1	•	o	o	o	o		o		•	•	•	•	•	•	•	
I2	•	o	o	o	o		o		•	•	•	•	•	•	•	
ok	•															•
Q	•	o	o	o	o		o		•	•	•	•	•	•		

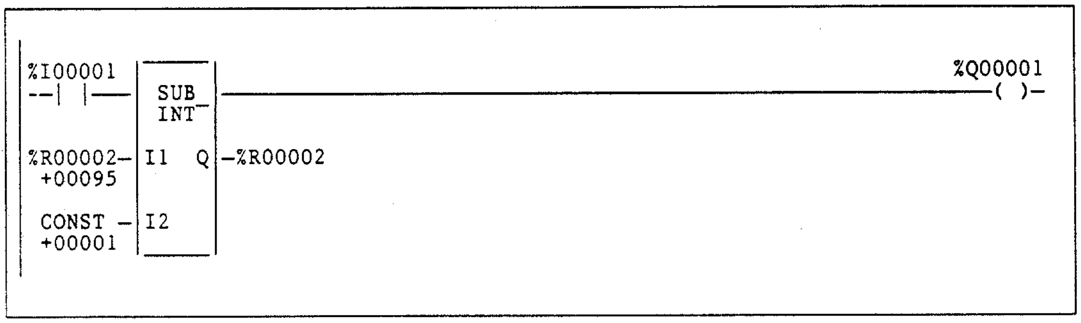
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for UINT or INT data only; not valid for DINT or REAL.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, whenever input %I00001 is set, the integer content of %R00002 is decremented by 1 and coil %Q00001 is turned on, provided there is no overflow in the subtraction.



MOD (INT, UINT, DINT)

The Modulo (MOD) function is used to divide one value by another value of the same data type, to obtain a remainder. If signed or double precision signed integers are used, the sign of the result is always the same as the sign of I1.

The MOD function operates on these types of data:

Data Type	Description
INT	Signed integer.
DINT	Double precision signed integer.
UINT	Unsigned integer.

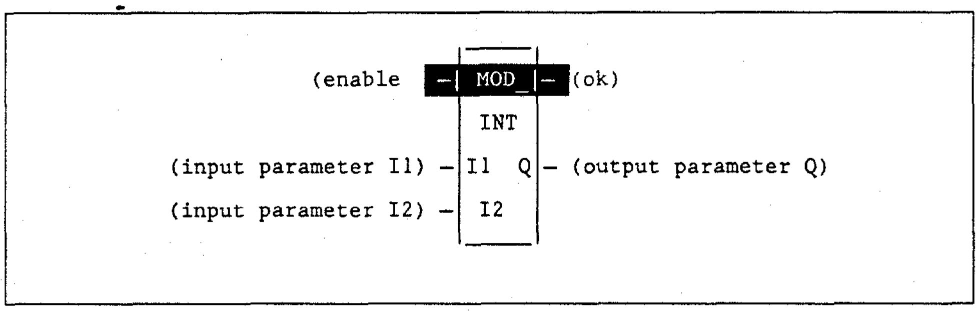
The default data type is signed integer; however, it can be changed after selecting the function. For more information on data types, refer to chapter 2, section 3, "Program Organization and User Data."

When the function receives power flow, it divides input parameter I1 by input parameter I2. These parameters must be the same data type. Output Q is calculated using the formula:

```
Q = I1 - ((I1 DIV I2) * I2)
```

where DIV produces an integer number. Q is the same data type as input parameters I1 and I2.

OK is always ON when the function receives power flow, unless there is an attempt to divide by zero. In that case, it is set OFF.



Parameters:

Parameter	Description
enable	When the function is enabled, the operation is performed.
I1	I1 contains a constant or reference for the value to be divided by I2.
I2	I2 contains a constant or reference for the value to be divided into I1.
ok	The ok output is energized when the function is performed without dividing by zero.
Q	Output Q contains the result of dividing I1 by I2 to obtain a remainder.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
I1	•	o	o	o	o		o		•	•	•	•	•	•	•	
I2	•	o	o	o	o		o		•	•	•	•	•	•	•	
ok	•															•
Q	•	o	o	o	o		o		•	•	•	•	•	•		

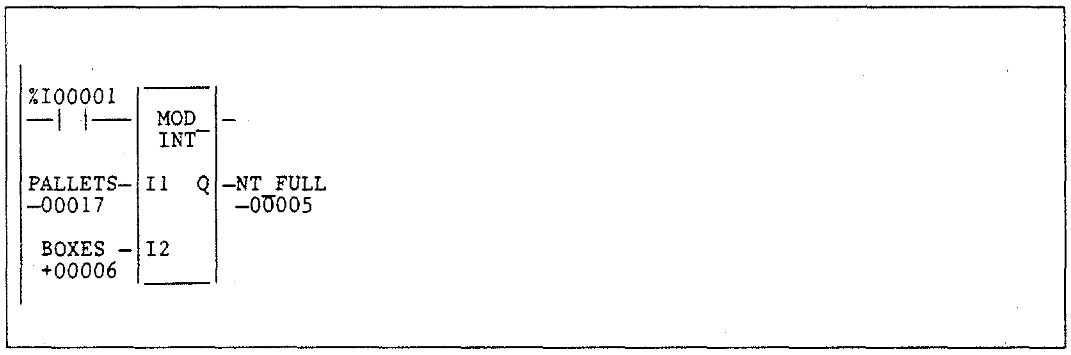
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for UINT or INT data only; not valid for DINT.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, the remainder of the integer division of BOXES into PALLETS is placed into NT_FULL whenever %I00001 is ON.



SQRT (INT, DINT, REAL)

The Square Root (SQRT) function is used to find the square root of a value. When the function receives power flow, the value of output Q is set to the integer portion of the square root of the input IN. The output Q must be the same data type as IN.

The SQRT function operates on these types of data:

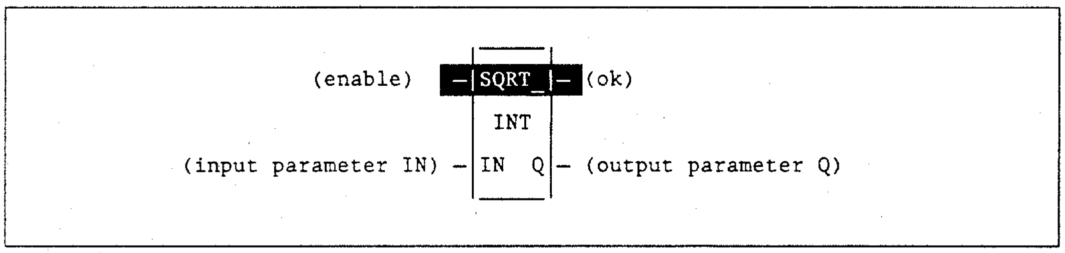
Data Type	Description
INT	Signed integer.
DINT	Double precision signed integer
REAL	Floating point.

The default data type is signed integer; however, it can be changed after selecting the function. For more information on data types, refer to chapter 2, section 3, "Program Organization and User Data."

OK is set ON if the function is performed without overflow, unless one of these invalid REAL operations occurs:

- IN < 0.
- IN is NaN (Not a Number).

Otherwise, ok is set OFF.



Parameters:

Parameter	Description
enable	When the function is enabled, the operation is performed.
IN	IN contains a constant or reference for the value whose square root is to be calculated. If IN is less than zero, the function will not pass power flow.
ok	The ok output is energized when the function is performed without overflow, unless an invalid operation occurs or IN is NaN.
Q	Output Q contains the square root of IN.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
I1	•	o	o	o	o		o		•	•	•	•	•	•	•	
ok	•															•
Q	•	o	o	o	o		o		•	•	•	•	•	•		

Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for INT only; not valid for DINT or REAL.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, the square root of the integer number located at %AI0001 is placed into the result located at %R00003 whenever %I00001 is ON.

```
|
|%I00001  _______
|—| |——|SQRT_  |
|        | INT‾ |
|        |      |
|%AI0001-|IN   Q|-%R00003
|        |______|
|
```

ABS (INT, DINT, REAL)

The Absolute Value (ABS) function is used to find the absolute value of an integer, double precision integer, or real value. When the function receives power flow, it places the absolute value of input IN in output Q.

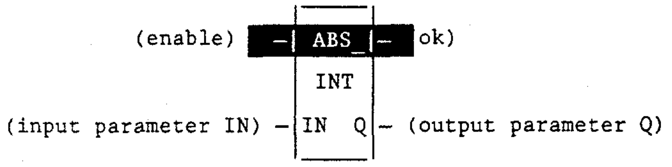
The ABS function operates on these types of data:

Data Type	Description
INT	Signed integer.
DINT	Double precision signed integer
REAL	Floating point.

The default data type is signed integer; however, it can be changed after selecting the function. For more information on data types, refer to chapter 2, section 3, "Program Organization and User Data."

The ok output will receive power flow, unless one of the following conditions occurs:

- For INT type, IN is MININT.
- For DINT type, IN is MINDINT.
- For REAL type, IN is NaN (Not a Number).



Parameters:

Parameter	Description
enable	When the function is enabled, the operation is performed.
IN	IN contains the integer or real value to be operated on.
ok	The ok output is energized when the function is performed without overflow, unless an invalid operation occurs and/or IN is NaN.
Q	Output Q contains the absolute value of IN.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
IN	•	o	o	o	o		o		•	•	•	•	•	•	•	
ok	•															•
Q	•	o	o	o	o		o		•	•	•	•	•	•		

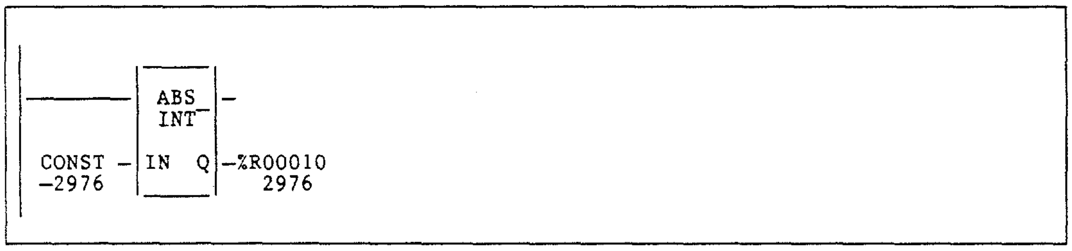
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for INT only; not valid for DINT or REAL.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, the absolute value of −2976, which is 2976, is placed in %R00010.



Trig Functions (SIN, COS, TAN, ASIN, ACOS, ATAN)

The SIN, COS, and TAN functions are used to find the trigonometric sine, cosine, and tangent, respectively, of its input. When one of these functions receives power flow, it computes the sine (or cosine or tangent) of IN, whose units are radians, and stores the result in output Q. Both IN and Q are floating-point values.

The ASIN, ACOS, and ATAN functions are used to find the inverse sine, cosine, and tangent, respectively, of its input. When one of these functions receives power flow, it computes the inverse sine (or cosine or tangent) of IN and stores the result in output Q, whose units are radians. Both IN and Q are floating-point values.

The SIN, COS, and TAN functions accept a broad range of input values, where $-2^{63} < \text{IN} < +2^{63}$, $(2^{63} \approx 9.22\text{x}10^{18})$.

The ASIN and ACOS functions accept a narrow range of input values, where $- \leq \text{IN} \leq 1$. Given a valid value for the IN parameter, the ASIN_REAL function will produce a result Q such that:

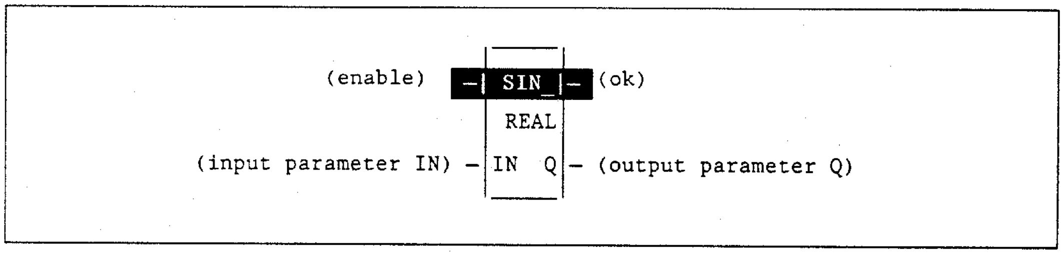
$$\text{ASIN (IN)} \; = \; -\frac{\pi}{2} \leq Q \leq \frac{\pi}{2}$$

The ACOS_REAL function will produce a result Q such that:

$$\text{ACOS (IN)} \; = \; - \; 0 \leq Q \leq \pi$$

The ATAN function accepts the broadest range of input values, where $-\infty \leq \text{IN} \leq +\infty$. Given a valid value for the IN parameter, the ATAN_REAL function will produce a result Q such that:

$$\text{ATAN (IN)} \; = \; -\frac{\pi}{2} \leq Q \leq \frac{\pi}{2}$$



Note

These functions are only available on floating-point versions of the Series 90-70 CPU.

Parameters:

Parameter	Description
enable	When the function is enabled, the operation is performed.
IN	IN contains the real value to be operated on.
ok	The ok output is energized when the function is performed without overflow, unless an invalid operation occurs and/or IN is NaN.
Q	Output Q contains the trigonometric value of IN.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
IN	•								•	•	•	•	•	•	•	
ok	•															•
Q	•								•	•	•	•	•	•		

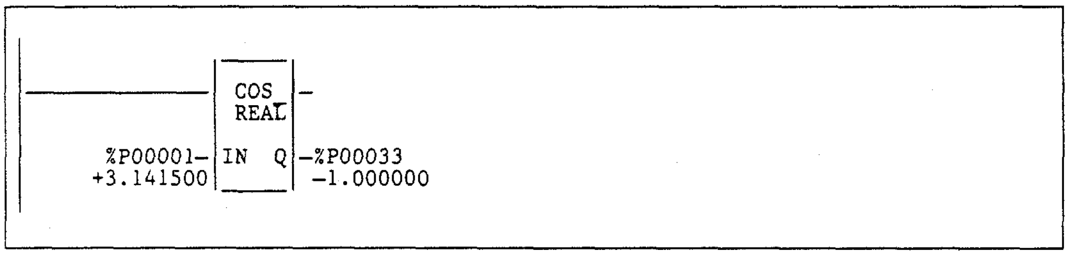
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, the COS of the value in %P00001 is placed in %P00033.



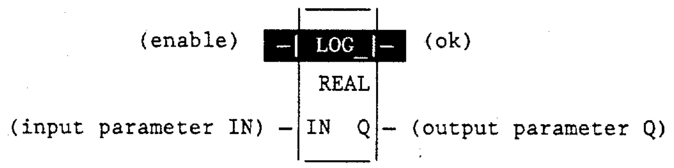
Logarithmic/Exponential Functions (LOG, LN, EXP, EXPT)

The LOG, LN, and EXP functions have two input parameters and two output parameters. When the function receives power flow, it performs the appropriate logarithmic/exponential operation on the real value in input IN and places the result in output Q.

- For the LOG function, the base 10 logarithm of IN is placed in Q.
- For the LN function, the natural logarithm of IN is placed in Q.
- For the EXP function, *e* is raised to the power specified by IN and the result is placed in Q.

The EXPT function has three input parameters and two output parameters. When the function receives power flow, the value of input I1 is raised to the power specified by the value I2 and the result is placed in output Q.

The ok output will receive power flow, unless IN is NaN (Not a Number) or is negative.



Parameters:

Parameter	Description
enable	When the function is enabled, the operation is performed.
IN	IN contains the real value to be operated on.
ok	The ok output is energized when the function is performed without overflow, unless an invalid operation occurs and/or IN is NaN or is negative.
Q	Output Q contains the logarithmic/exponential value of IN.

Note

These functions are only available on floating-point versions of the Series 90-70 CPU.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
IN*	•								•	•	•	•	•	•	•	
ok	•															•
Q	•								•	•	•	•	•	•		

Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
* For the EXPT function, input IN is replaced by input parameters I1 and I2.
• Valid reference or place where power may flow through the function.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

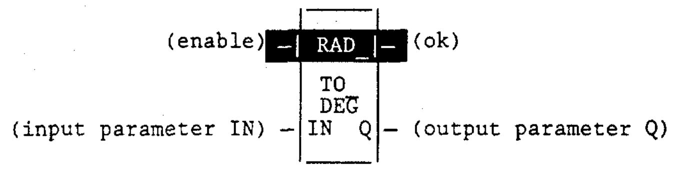
In the following example, the value of %AI0001 is raised to the power of 2.5 and the result is placed in %R00001.

```
|
|              _______
|_____________|EXPT   |-
|             | REAL  |
|             |       |
|   %AI0001-|I1    Q|-%R00001
|             |       |
|     CONST  -|I2     |
| 2.50000E+00|_______|
|
```

Radian Conversion (RAD, DEG)

When the function receives power flow, the appropriate conversion (RAD_TO_DEG or DEG_TO_RAD) is performed on the real value in input IN and the result is placed in output Q.

The ok output will receive power flow unless IN is NaN (Not a Number).



Parameters:

Parameter	Description
enable	When the function is enabled, the operation is performed.
IN	IN contains the real value to be operated on.
ok	The ok output is energized when the function is performed without overflow, unless IN is NaN.
Q	Output Q contains the converted value of IN.

Note

These functions are only available on floating-point versions of the Series 90-70 CPU.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
IN	•								•	•	•	•	•	•	•	
ok	•															•
Q	•								•	•	•	•	•	•		

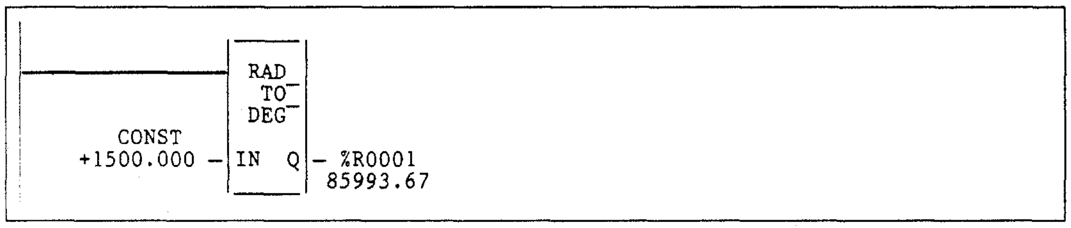
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, +1500 is converted to DEG and is placed in %R0001.



Section 4: Relational Functions

Relational functions are used to compare two numbers. This section describes the following relational functions:

Abbreviation	Function	Description	Page
EQ	Equal	Test two numbers for equality.	4-47
NE	Not Equal	Test two numbers for non-equality.	4-47
GT	Greater Than	Test for one number greater than another.	4-47
GE	Greater Than or Equal	Test for one number greater than or equal to another.	4-47
LT	Less Than	Test for one number less than another.	4-47
LE	Less Than or Equal	Test for one number less than or equal to another.	4-47
CMP	Compare	Test for one number less than, equal to, or greater than another.	4-49
RANGE	Range	Determine whether a number is within a specified range.	4-51

EQ, NE, GT, GE, LT, and LE (INT, UINT, DINT, REAL)

Relational functions are used to determine the relation of two values. When the function receives power flow, it compares input parameter I1 to input parameter I2. These parameters must be the same data type.

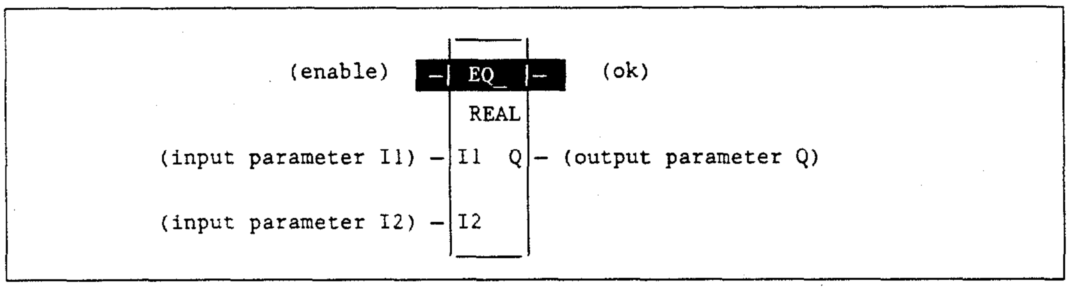
Relational functions operate on these types of data:

Data Type	Description
INT	Signed integer.
UINT	Unsigned integer.
DINT	Double precision signed integer.
REAL	Floating point.

The default data type is signed integer. To compare either signed integer, unsigned integer, or double precision integers, select the new data type after selecting the relational function. To compare data of other types or of two different types, first use the appropriate conversion function (described in section 8, *Conversion Functions*) to change the data to one of the integer types.

If input parameters I1 and I2 match the specified relation, output Q receives power flow and is set ON (1); otherwise, it is set OFF (0).

Output ok will always receive power flow when the function is enabled, unless I1 and/or I2 is NaN (Not a Number).



Parameters:

Parameter	Description
enable	When the function is enabled, the operation is performed.
I1	I1 contains a constant or reference for the first value to be compared. (I1 is on the left side of the relational equation, as in I1 < I2).
I2	I2 contains a constant or reference for the second value to be compared. (I2 is on the right side of the relational equation, as in I1 < I2).
ok	The ok output is energized when the function is performed without error, unless I1 and/or I2 is NaN.
Q	Output Q is energized when I1 and I2 match the specified relation.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
I1	•	o	o	o	o		o		•	•	•	•	•	•	•	
I2	•	o	o	o	o		o		•	•	•	•	•	•	•	
ok	•															•
Q	•	o	o	o	o		o		•	•	•	•	•	•		

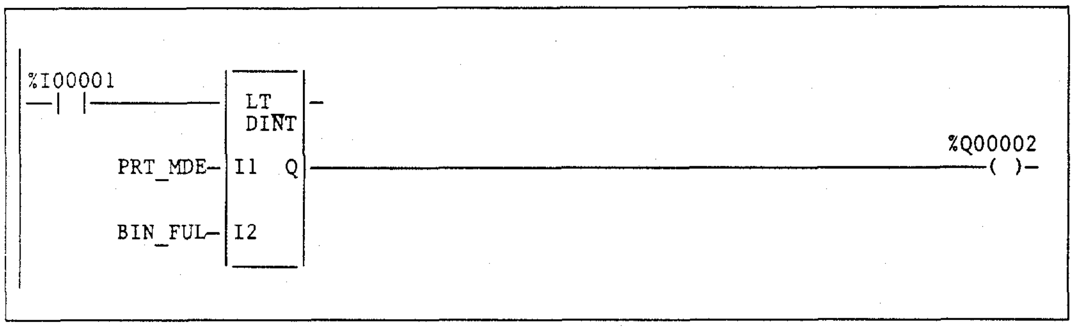
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for INT or UINT data only; not valid for DINT or REAL.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, two double precision signed integers, PRT_MDE and BIN_FUL, are compared whenever %I00001 is set. If PRT_MDE is less than BIN_FUL, coil %Q00002 is turned on.



CMP (INT, UINT, DINT, REAL)

Use the Compare (CMP) function to test for one number less than, equal to, or greater than another.

When the function receives power flow, it compares the value I1 to the value I2. These values must be the same data type.

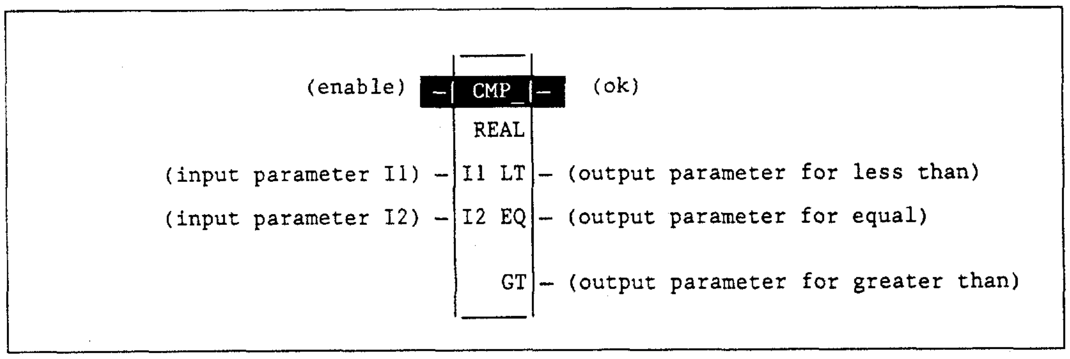
The CMP function operates on these types of data:

Data Type	Description
INT	Signed integer.
UINT	Unsigned integer.
DINT	Double precision signed integer.
REAL	Floating point.

The default data type is signed integer. To compare either signed integer, unsigned integer, or double precision signed integers, select the new data type after selecting the relational function. To compare data of other types or of two different types, first use the appropriate conversion function (described in section 8, *Conversion Functions*) to change the data to one of the integer types.

LT, EQ, and GT are set ON (1) or OFF (0) as a result of the comparison.

Output ok will always receive power flow when the function is enabled, unless I1 and/or I2 is NaN (Not a Number).


(enable)
CMP_
(ok)
REAL
(input parameter I1)
I1
LT
(output parameter for less than)
(input parameter I2)
I2
EQ
(output parameter for equal)
GT
(output parameter for greater than)

Parameters:

Parameter	Description
enable	When the function is enabled, the operation is performed.
I1	I1 contains a constant or reference for the first value to be compared.
I2	I2 contains a constant or reference for the second value to be compared.
ok	The ok output is energized when the function is performed without error, unless I1 and/or I2 is NaN.
LT	Output LT is energized when I1 is less than I2.
EQ	Output EQ is energized when I1 is equal to I2.
GT	Output GT is energized when I1 is greater than I2.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
I1	•	o	o	o	o		o		•	•	•	•	•	•	•	
I2	•	o	o	o	o		o		•	•	•	•	•	•	•	
ok	•															•
LT	•															•
EQ	•															•
GT	•															•

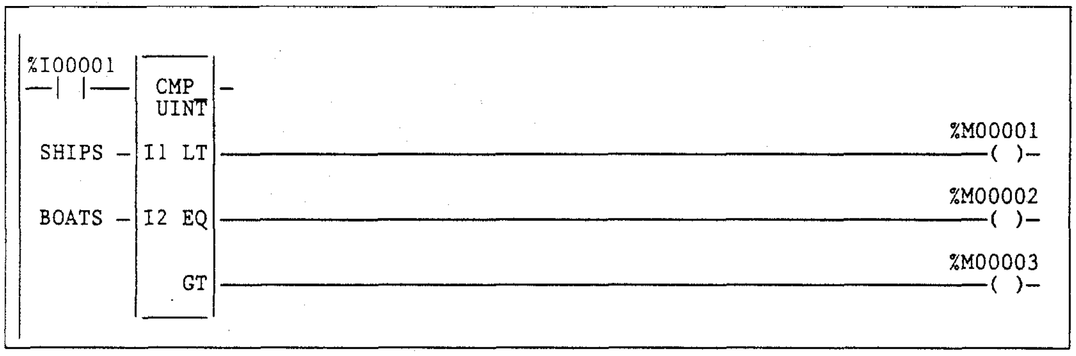
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for INT or UINT data only; not valid for DINT or REAL.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, when %I00001 is ON, the integer variable SHIPS is compared with the variable BOATS. Internal coils %M0001, %M0002, and %M0003 are set to the results of the compare.



RANGE (INT, DINT, WORD, DWORD)

The RANGE function is used to compare a single input value against two delimiters to determine whether the input value falls within the range of the delimiters.

Note

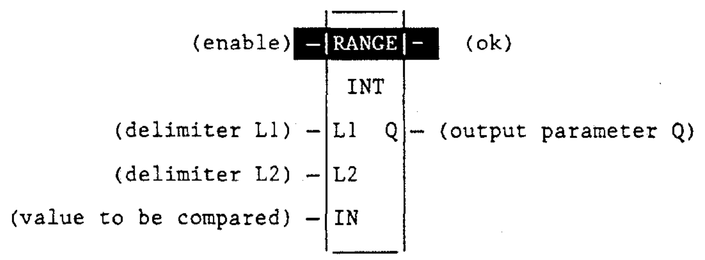
The Range function is only available on a Release 5 or higher CPU.

The RANGE function operates on these types of data:

Data Type	Description
INT	Signed integer.
DINT	Double precision signed integer.
UINT	Unsigned integer.
WORD	Word data type.
DWORD	Double word data type.

The default data type is signed integer; however, it can be changed after selecting the function. For more information on data types, please refer to the "Data Types" section on page 2-16.

When the function is enabled, the RANGE function block will compare the value in input parameter IN against the range specified by the values of the two delimiters specified by parameters L1 and L2, inclusively. When the value in IN is within the range specified by L1 and L2, output parameter Q is set ON (1). Otherwise, Q is set OFF (0). If the operation is successful, the ok output will receive power flow.



Note

Limit parameters L1 and L2 represent the endpoints of a range. There is no fixed minimum/maximum or high/low connotation assigned to either parameter. Therefore, a desired range of 0 to 100 could be specified by assigning 0 to L1 and 100 to L2 or 0 to L2 and 100 to L1. In either case, a value of 45 for the input parameter IN would result in the output Q being set ON (1).

Parameters:

Parameter	Description
enable	When the function is enabled, the operation is performed.
L1	L1 contains the start point of the range.
L2	L2 contains the end point of the range.
IN	IN contains the value to be compared against the range specified by L1 and L2.
ok	The ok output is energized unless an error is encountered.
Q	Output Q is energized when the value in IN is within the range specified by L1 and L2, inclusive.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
L1		o	o	o	o		o		•	•	•	•	•		•‡	
L2		o	o	o	o		o		•	•	•	•	•		•‡	
IN		o	o	o	o		o		•	•	•	•	•			
ok		o	o	o	o		o		•	•	•	•	•			
Q	•															•

Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for INT or WORD data only; not valid for DINT or DWORD.
‡ Constants are limited to integer values for double precision signed integer operations.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, the value in %R00001 is 10, and the value in %R00002 is 50. The output Q will be ON for all IN values in %R00003 greater than or equal to 10 and less than or equal to 50. Output Q will be OFF for all IN values in %R00003 less than 10 or greater than 50. The ok output is set ON.

```
|
|%I00001 |‾‾‾‾‾‾‾|                                                    %M00001
|--| |---+RANGE |---------------------------------------------------( )-
|        | INT  |
|        |      |                                                    %M00002
|%R00001-|L1   Q|---------------------------------------------------( )-
|        |      |
|%R00002-|L2    |
|        |      |
|%R00003-|IN    |
|        |______|
|
```

Section 5: Bit Operation Functions

Bit operation functions perform comparison, logical, and move operations on bit strings. The maximum string length is 256 words or double words. Bit operation functions require WORD or DWORD data; the default data type is WORD.

Although data must be specified in 16-bit or 32-bit increments, these functions operate on data as a continuous string of bits, with bit 1 of the first word being the Least Significant Bit (LSB). The last bit of the last word is the Most Significant Bit (MSB). For example, if you specified three words of data beginning at reference %L00100, it would be operated on as 48 contiguous bits.

%L00100	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	← bit 1 (LSB)
%L00101	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	
%L00102	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	
	↑ (MSB)																

Note

Overlapping input and output reference address ranges in multi-word functions may produce unexpected results.

The following bit operation functions are described in this section:

Abbreviation	Function	Description	Page
AND	Logical AND	If a bit in bit string I1 and the corresponding bit in bit string I2 are both 1, place a 1 in the corresponding location in output string Q.	4-55
OR	Logical OR	If a bit in bit string I1 and/or the corresponding bit in bit string I2 are both 1, place a 1 in the corresponding location in output string Q.	4-55
XOR	Logical exclusive OR	If a bit in bit string I1 and the corresponding bit in string I2 are different, place a 1 in the corresponding location in the output bit string.	4-57
NOT	Logical Invert	Set the state of each bit in output bit string Q to the opposite state of the corresponding bit in bit string I1.	4-59
SHL	Shift Left	Shift all the bits in a word or string of words to the left by a specified number of places.	4-61
SHR	Shift Right	Shift all the bits in a word or string of words to the right by a specified number of places.	4-61
ROL	Rotate Left	Rotate all the bits in a string a specified number of places to the left.	4-64
ROR	Rotate Right	Rotate all the bits in a string a specified number of places to the right.	4-64
BTST	Bit Test	Test a bit within a bit string to determine whether that bit is currently 1 or 0.	4-66
BSET	Bit Set	Set a bit in a bit string to 1.	4-68
BCLR	Bit Clear	Clear a bit within a string by setting that bit to 0.	4-68
BPOS	Bit Position	Locate a bit set to 1 in a bit string.	4-70
MCMP	Masked Compare	Compare the bits in the first string with the corresponding bits in the second.	4-72

Note

Note that, for all bit operations, the bit group of function blocks not explicitly bit-typed will affect the transitions (coils and contacts) for all bits in the written byte/word/dword. Please read the second example shown on page 4-69 for further explanation.

AND and OR (WORD, DWORD)

Each scan that power is received, the AND or OR function examines each bit in bit string I1 and the corresponding bit in bit string I2, beginning at the least significant bit in each.

For each two bits examined for the AND function, if both bits are 1, then a 1 is placed in the corresponding location in output string Q. If either or both bits are 0, then a 0 is placed in string Q in that location.

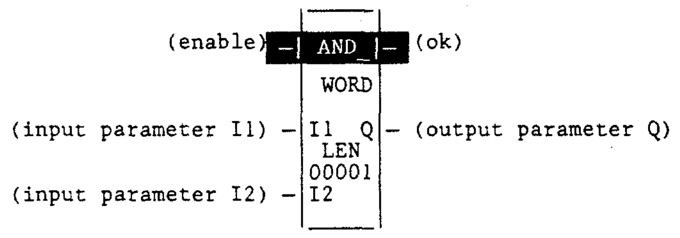
The AND function is useful for building masks or screens, where only certain bits are passed through (those that are opposite a 1 in the mask), and all other bits are set to 0. The function can also be used to clear the selected area of word memory by ANDing the bits with another bit string known to contain all 0s. The I1 and I2 bit strings specified may overlap.

For each two bits examined for the OR function, if either or both bits are 1, then a 1 is placed in the corresponding location in output string Q. If both bits are 0, then a 0 is placed in string Q in that location.

The OR function is useful for combining strings, and to control many outputs through the use of one simple logical structure. The function is the equivalent of two relay contacts in parallel multiplied by the number of bits in the string. It can be used to drive indicator lamps directly from input states, or superimpose blinking conditions on status lights.

The string length can be up to 256 words or double words for either function.

The function passes power flow to the right whenever power is received.



Parameters:

Parameter	Description
enable	When the function is enabled, the operation is performed.
I1	I1 contains a constant or reference for the first word.
I2	I2 contains a constant or reference for the second word.
ok	The ok output is energized whenever enable is energized.
Q	Output Q contains the result of the operation.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
I1	•	o	o	o	o	o	o		•	•	•	•	•	•	•	
I2	•	o	o	o	o	o	o		•	•	•	•	•	•	•	
ok	•															•
Q	•	o	o	o	o	o†	o		•	•	•	•	•	•		

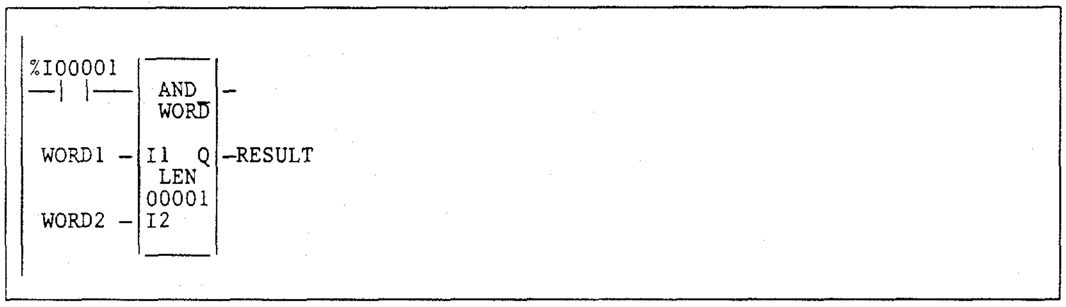
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for WORD data only; not valid for DWORD.
† %SA, %SB, %SC only; %S cannot be used.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, whenever input %I00001 is set, the 16-bit strings represented by nicknames WORD1 and WORD2 are examined. The results of the logical AND are placed in output string RESULT.



WORD1	0	0	0	1	1	1	1	1	1	1	0	0	1	0	0	0
WORD2	1	1	0	1	1	1	0	0	0	0	0	0	1	1	1	1

RESULT	0	0	0	1	1	1	0	0	0	0	0	0	1	0	0	0

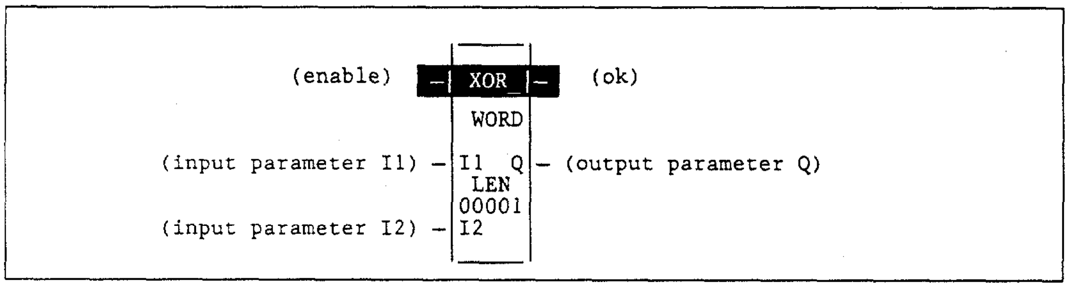
XOR (WORD, DWORD)

The Exclusive OR (XOR) function is used to compare each bit in bit string I1 with the corresponding bit in string I2. If the bits are different, a 1 is placed in the corresponding position in the output bit string.

Each scan that power is received, the function examines each bit in string I1 and the corresponding bit in string I2, beginning at the least significant bit in each. For each two bits examined, if only one is 1, then a 1 is placed in the corresponding location in bit string Q. The bit string length can be up to 256 words or double words. The XOR function passes power flow to the right whenever power is received.

If string I2 and output string Q begin at the same reference, a 1 placed in string I1 will cause the corresponding bit in string I2 to alternate between 0 and 1, changing state with each scan as long as power is received. Longer cycles can be programmed by pulsing the power flow to the function at twice the desired rate of flashing; the power flow pulse should be one scan long (one-shot type coil or self-resetting timer).

The XOR function is useful for quickly comparing two bit strings, or to blink a group of bits at the rate of one ON state per two scans.



Parameters:

Parameter	Description
enable	When the function is enabled, the operation is performed.
I1	I1 contains a constant or reference for the first word to be XORed.
I2	I2 contains a constant or reference for the second word to be XORed.
ok	The ok output is energized whenever enable is energized.
Q	Output Q contains the result of I1 XORed with I2.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
I1	•	o	o	o	o	o	o		•	•	•	•	•	•	•	
I2	•	o	o	o	o	o	o		•	•	•	•	•	•	•	
ok	•															•
Q	•	o	o	o	o	o†	o		•	•	•	•	•	•		

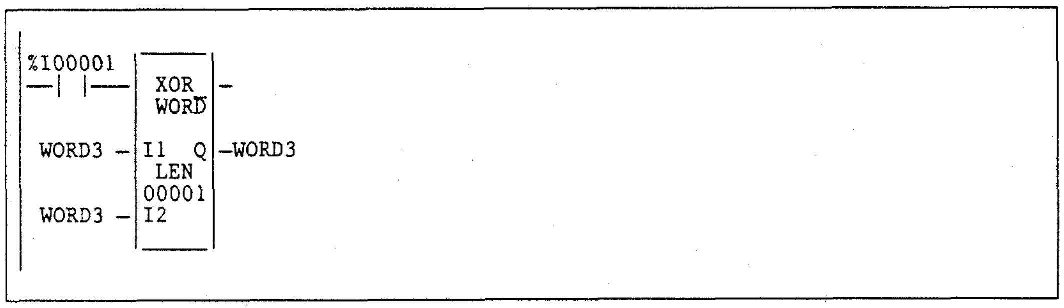
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for WORD data only; not valid for DWORD.
† %SA, %SB, %SC only; %S cannot be used.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, whenever input %I00001 is set, the 16-bit string represented by the nickname WORD3 is cleared (set to all zeros).



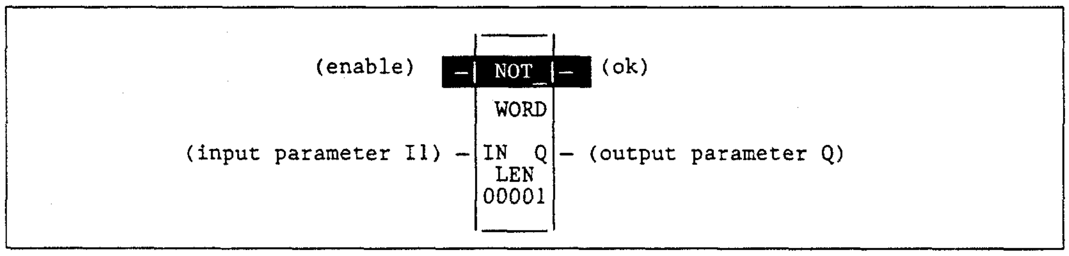
I1 (WORD3)	0	0	0	1	1	1	1	1	1	1	0	0	1	0	0	0
I2 (WORD3)	0	0	0	1	1	1	1	1	1	1	0	0	1	0	0	0

Q (WORD3)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NOT (WORD, DWORD)

The NOT function is used to set the state of each bit in the output bit string Q to the opposite of the state of the corresponding bit in bit string I1.

All bits are changed on each scan that power is received, making output string Q the logical complement of I1. A length of 1 to 256 words or double words can be selected. The function passes power flow to the right whenever power is received.



Parameters:

Parameter	Description
enable	When the function is enabled, the operation is performed.
I1	I1 contains a constant or reference for the word to be negated.
ok	The ok output is energized whenever enable is energized.
Q	Output Q contains the NOT (negation) of I1.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
I1	•	o	o	o	o	o	o		•	•	•	•	•	•	•	
ok	•															•
Q	•	o	o	o	o	o†	o		•	•	•	•	•	•		

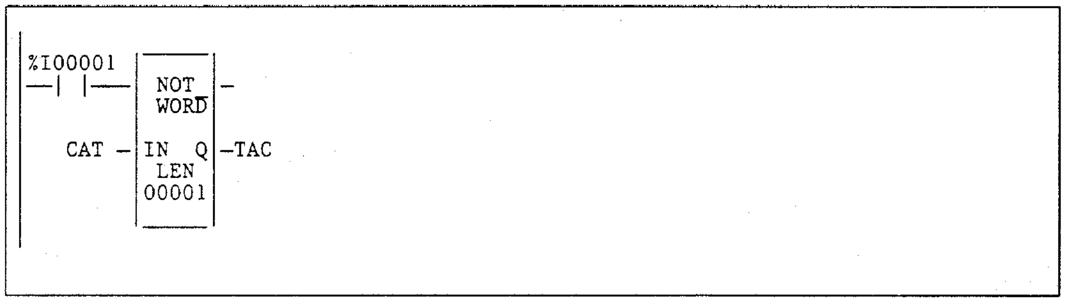
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for WORD data only; not valid for DWORD.
† %SA, %SB, %SC only; %S cannot be used.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

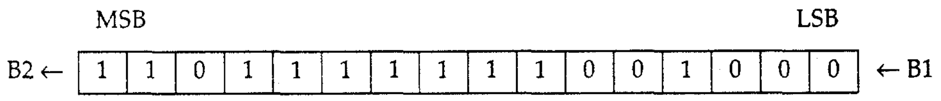
Example:

In the following example, whenever input %I00001 is set, the bit string represented by the nickname TAC is set to the inverse of bit string CAT.

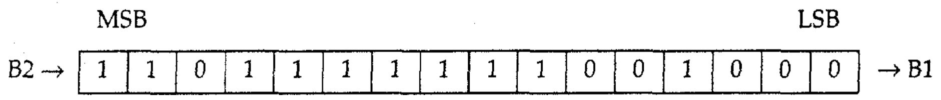


SHL and SHR (WORD, DWORD)

The Shift Left (SHL) function is used to shift all the bits in a word or group of words to the left by a specified number of places. When the shift occurs, the specified number of bits is shifted out of the output string to the left. As bits are shifted out of the high end of the string, the same number of bits is shifted in at the low end.



The Shift Right (SHR) function is used to shift all the bits in a word or group of words a specified number of places to the right. When the shift occurs, the specified number of bits is shifted out of the output string to the right. As bits are shifted out of the low end of the string, the same number of bits is shifted in at the high end.



A string length of 1 to 256 words or double words can be selected for either function.

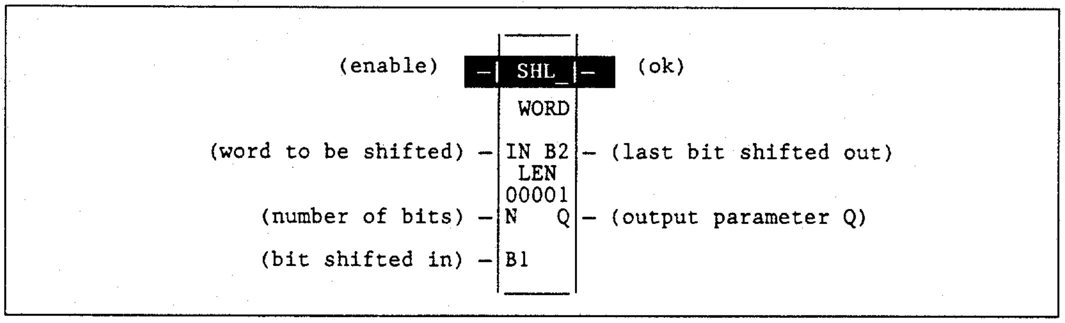
The number of places specified for the shift must be more than zero and less than the number of bits in the string. Otherwise, no shift occurs and no power flow is generated.

The bits being shifted into the beginning of the string are specified via input parameter B1. If a length greater than 1 has been specified as the number of bits to be shifted, each of the bits is filled with the same value (0 or 1). This can be:

- The boolean output of another program function.
- All 1s. To do this, use the special reference nickname ALW_ON as a permissive to input B1.
- All 0s. To do this, use the special reference nickname ALW_OFF as a permissive to input B1.

The SHL or SHR function passes power flow to the right, unless the number of bits specified to be shifted is greater than the total length of the string or is zero.

Output Q is the shifted copy of the input string. If you want the input string to be shifted, the output parameter Q must use the same memory location as the input parameter IN. The entire shifted string is written on each scan that power is received. Output B2 is the last bit shifted out. For example, if four bits were shifted, B2 would be the fourth bit shifted out.



Parameters:

Parameter	Description
enable	When the function is enabled, the shift is performed.
IN	IN contains the first word to be shifted.
N	N contains the number of places (bits) that the array is to be shifted.
B1	B1 contains the bit value to be shifted into the array.
B2	B2 contains the bit value of the last bit shifted out of the array.
ok	The ok output is energized when the shift is energized and the shift length is not greater than the array size.
Q	Output Q contains the first word of the shifted array.
LEN	LEN is the number of words in the array to be shifted.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
IN	•	o	o	o	o	o	o		•	•	•	•	•	•	•	
N	•	•	•	•	•		•		•	•	•	•	•	•	•	
B1	•															
B2	•															•
ok	•															•
Q	•	o	o	o	o	o†	o		•	•	•	•	•	•		

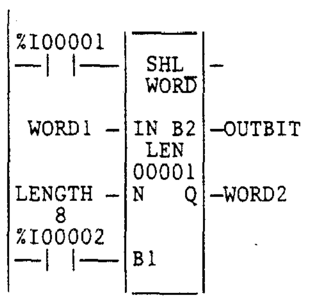
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for WORD data only; not valid for DWORD.
† %SA, %SB, %SC only; %S cannot be used.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, whenever input %I00001 is set, the output bit string represented by the nickname WORD2 is made a copy of WORD1, left-shifted by the number of bits represented by the nickname LENGTH. The resulting open bits at the beginning of the output string are set to the value of %I00002.



ROL and ROR (WORD, DWORD)

The Rotate Left (ROL) function is used to rotate all the bits in a string a specified number of places to the left. When rotation occurs, the specified number of bits is rotated out of the input string to the left and back into the string on the right.

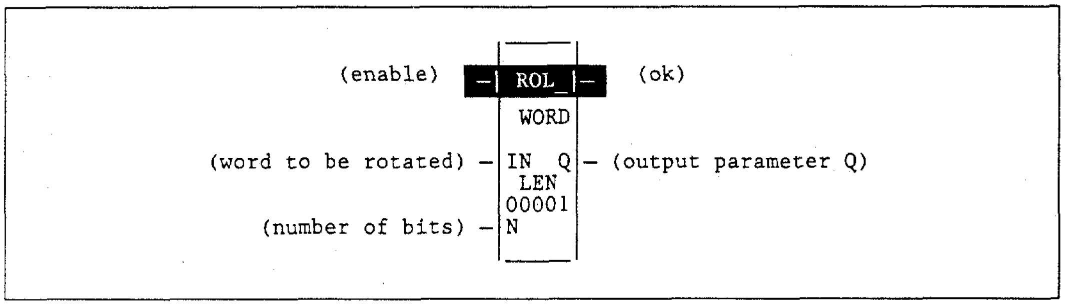
The Rotate Right (ROR) function rotates the bits in the string to the right. When rotation occurs, the specified number of bits is rotated out of the input string to the right and back into the string on the left.

A string length of 1 to 256 words or double words can be selected for either function.

The number of places specified for rotation must be more than zero and less than the number of bits in the string. Otherwise, no movement occurs and no power flow is generated.

The ROL or ROR function passes power flow to the right, unless the number of bits specified to be rotated is greater than or equal to the total length of the string or is less than or equal to zero.

The result is placed in output string Q. If you want the input string to be rotated, the output parameter Q must use the same memory location as the input parameter IN. The entire rotated string is written on each scan that power is received.



Parameters:

Parameter	Description
enable	When the function is enabled, the rotation is performed.
IN	IN contains the first word to be rotated.
N	N contains the number of places that the array is to be rotated.
ok	The ok output is energized when the rotation is energized and the rotation length is not greater than the array size.
Q	Output Q contains the first word of the rotated array.
LEN	LEN is the number of words in the array to be rotated.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
IN	•	o	o	o	o	o	o		•	•	•	•	•	•	•	
N	•	•	•	•	•		•		•	•	•	•	•	•	•	
ok	•															•
Q	•	o	o	o	o	o†	o		•	•	•	•	•	•		

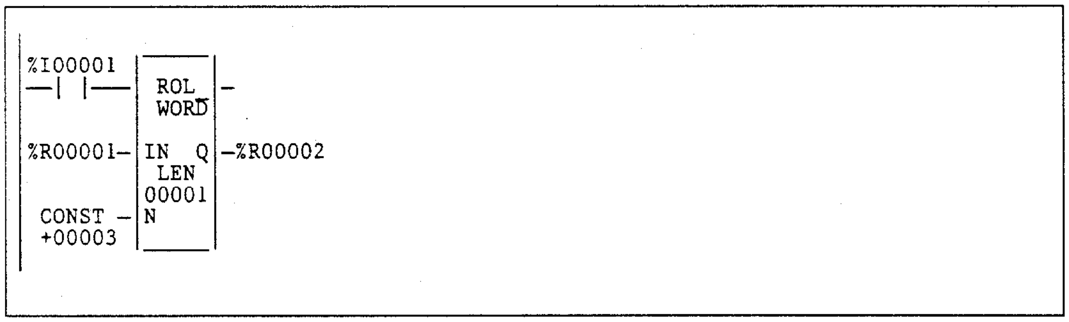
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for WORD data only; not valid for DWORD.
† %SA, %SB, %SC only; %S cannot be used.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, whenever input %I00001 is set, the input bit string %R00001 is rotated 3 bits and the result is placed in %R00002. After execution of this function, the input bit string %R00001 is unchanged. If the same reference is used for IN and Q, a rotation will occur in place.



<u>%R00001</u>:

	MSB															LSB	
←	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	←

<u>%R00002 (after %I00001 is set)</u>:

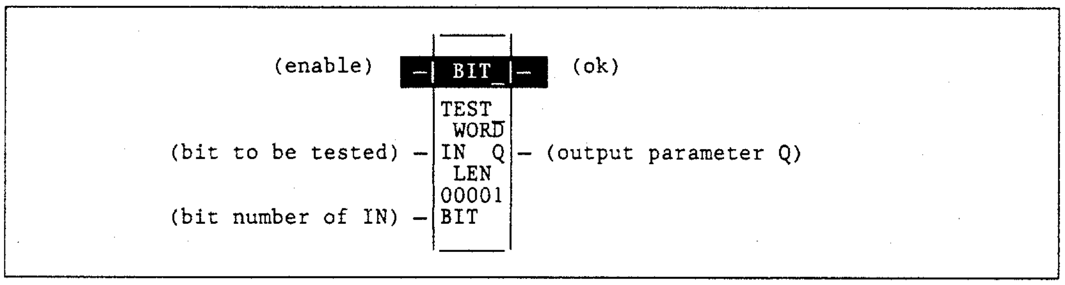
MSB															LSB
1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1

BTST (WORD, DWORD)

The Bit Test (BTST) function is used to test a bit within a bit string to determine whether that bit is currently 1 or 0. The result of the test is placed in output Q.

Each sweep power is received, the BTST function sets its output Q to the same state as the specified bit. If a register rather than a constant is used to specify the bit number, the same function block can test different bits on successive sweeps. If the value of BIT is outside the range (1 ≤ BIT ≥ (16 * LEN)), then Q is set OFF.

A string length of 1 to 256 words or double words can be selected.



Parameters:

Parameter	Description
enable	When the function is enabled, the bit test is performed.
IN	IN contains the first word of the data to be operated on.
BIT	BIT contains the bit number of IN that should be tested. Valid range is (1 ≤ BIT ≥ (16 * LEN)).
ok	The ok output is energized when enable is energized and BIT is greater than the string length or is zero.
Q	Output Q is energized if the bit tested was a 1.
LEN	LEN is the number of words in the string to be tested.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
IN	•	o	o	o	o	o	o		•	•	•	•	•	•	•	
BIT	•	•	•	•	•		•		•	•	•	•	•	•	•	
ok	•															•
Q	•															

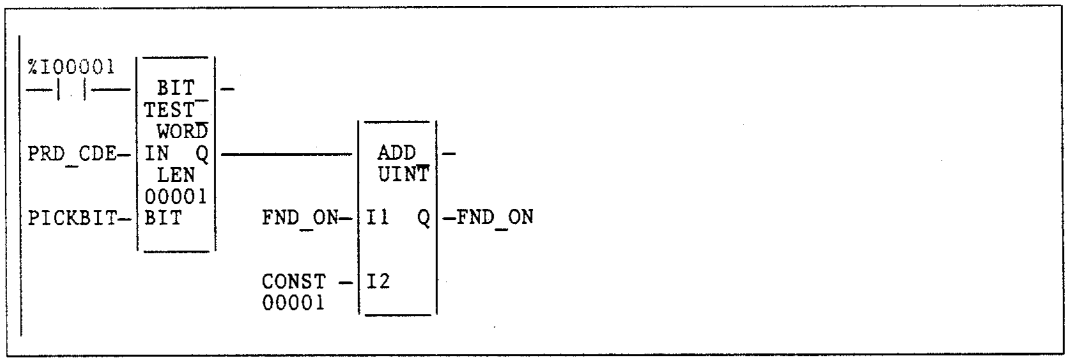
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for WORD data only; not valid for DWORD.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, whenever input %I00001 is set, the bit at the location contained in reference PICKBIT is tested. The bit is part of string PRD_CDE. If it is 1, output Q passes power flow to the ADD function, causing 1 to be added to the current value of the ADD function input I1.

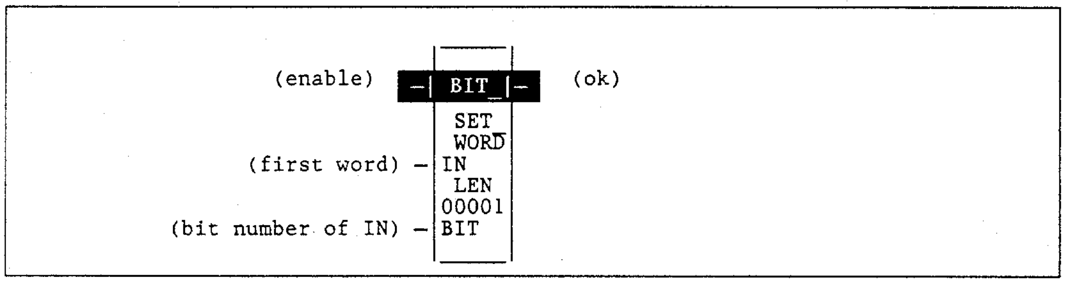


BSET and BCLR (WORD, DWORD)

The Bit Set (BSET) function is used to set a bit in a bit string to 1. The Bit Clear (BCLR) function is used to clear a bit within a string by setting that bit to 0.

Each sweep that power is received, the function sets the specified bit to 1 for the BSET function or to 0 for the BCLR function. If a variable (register) rather than a constant is used to specify the bit number, the same function block can set different bits on successive sweeps.

A string length of 1 to 256 words or double words can be selected. The function passes power flow to the right, unless the value for BIT is outside the range (1 ≤ BIT ≥ (16 * LEN)). Then, ok is set OFF.



Parameters:

Parameter	Description
enable	When the function is enabled, the bit operation is performed.
IN	IN contains the first word of the data to be operated on.
BIT	BIT contains the bit number of IN that should be set or cleared. Valid range is (1 ≤ BIT ≥ (16 * LEN)).
ok	The ok output is energized when enable is energized.
LEN	LEN is the number of words in the bit string.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
IN		o	o	o	o	o†	o		•	•	•	•	•	•		
BIT	•	•	•	•	•		•		•	•	•	•	•	•	•	
ok	•															•

Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for WORD data only; not valid for DWORD.
† %SA, %SB, %SC only; %S cannot be used.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34. Also, please note that, for all bit operations, the bit group of function blocks not explicitly bit-typed will affect the transitions (coils and contacts) for all bits in the written byte/word/dword. Please read the second example shown below and the explanation that precedes it before using bit operations with byte and word blocks that have transitions.

Example 1:

In the following example, whenever input %I00001 is set, bit 12 of the string beginning at reference %L00040 is set to 1.

```
|%I00001 |______|
|—| |——  | BIT  |-
|        | SET  |
|        | WORD |
|%L00040-|IN    |
|        | LEN  |
|        |00001 |
| CONST -|BIT   |
| 00012  |______|
|
```

Example 2:

In the following example, M41−M48 will be solved as written to be a Transition Status. **These bits may not perform as expected** when used as a transition contact or coil. If you wish to use Bit Op functions *in conjunction with transition functions*, your Bit Op function should be type BIT.

```
|%I00001 |______|
|—| |——  | BIT  |-
|        | SET  |
|        | WORD |
|%M00041-|IN    |
|        | LEN  |
|        |00001 |
| CONST -|BIT   |
| 00003  |______|
|
```

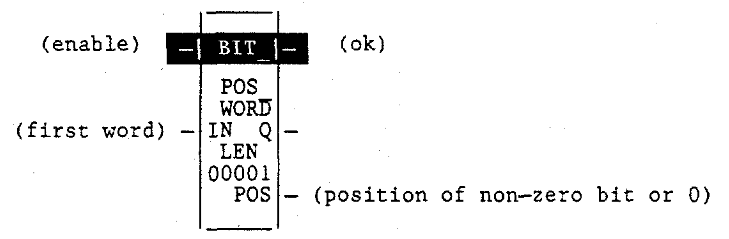
BPOS (WORD, DWORD)

The Bit Position (BPOS) function is used to locate a bit set to 1 in a bit string.

Each sweep that power is received, the function scans the bit string starting at IN. When the function stops scanning, either a bit equal to 1 has been found or the entire length of the string has been scanned.

POS is set to the position within the bit string of the first non-zero bit; POS is set to zero if no non-zero bit is found.

A string length of 1 to 256 words can be selected. The function passes power flow to the right whenever enable is ON.



Parameters:

Parameter	Description
enable	When the function is enabled, a bit search operation is performed.
IN	IN contains the first word of the data to be operated on.
ok	The ok output is energized when enable is energized.
POS	The position of the first non-zero bit found, or zero if a non-zero bit is not found.
Q	Output Q is energized if a bit set to 1 is found.
LEN	LEN is the number of words in the bit string.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
IN	•	o	o	o	o	o	o		•	•	•	•	•	•	•	•
POS	•	•	•	•	•		•		•	•	•	•	•	•		
ok	•															•
Q	•															•

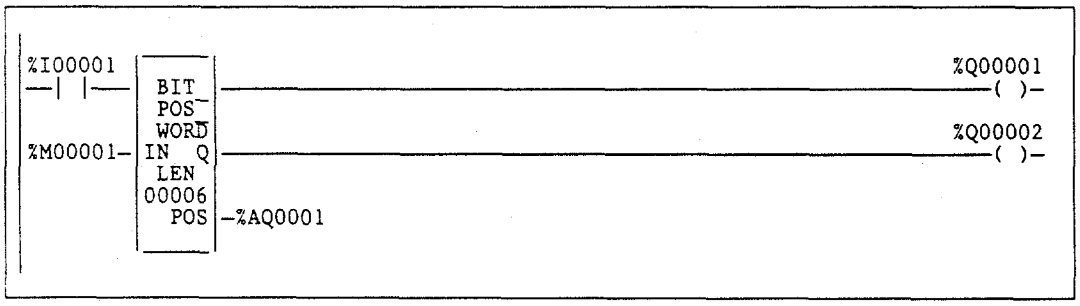
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for WORD data only; not valid for DWORD.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, if %I00001 is set, the bit string starting at %M00001 is searched until a bit equal to 1 is found, or 6 words have been searched. Coil %Q00001 is turned on. If a bit equal to 1 is found, its location within the bit string is written to %AQ0001 and %Q00002 is turned on. If %I00001 is set, bit %M00001 is 0, and bit %M00002 is 1, then the value written to %AQ0001 is 2.



MCMP (WORD, DWORD)

The Masked Compare (MCMP) function is used to compare the contents of two bit strings. Input string I1 might contain the states of outputs, such as solenoids or motor starters. Input string I2 might contain their input state feedback, such as limit switches or contacts.

Each scan that power is received, the function begins comparing the bits in the first string with the corresponding bits in the second. Comparison continues until a miscompare is found, or until the end of the string is reached.

The BIT input is used to store the bit number where the next comparison should start. Ordinarily, this is the same as the number where the last miscompare occurred. Because the bit number of the last miscompare is stored in output BN, the same reference can be used for both BIT and BN.

If you want to start the next comparison at some other location in the string, you can enter different references for BIT and BN. If the value of BIT is a location that is beyond the end of the string, BIT is reset to 1 before starting the next comparison.

The function passes power flow whenever power is received. The other outputs of the function depend on the state of the corresponding mask bit, as described below.

If All Bits in I1 and I2 are the Same

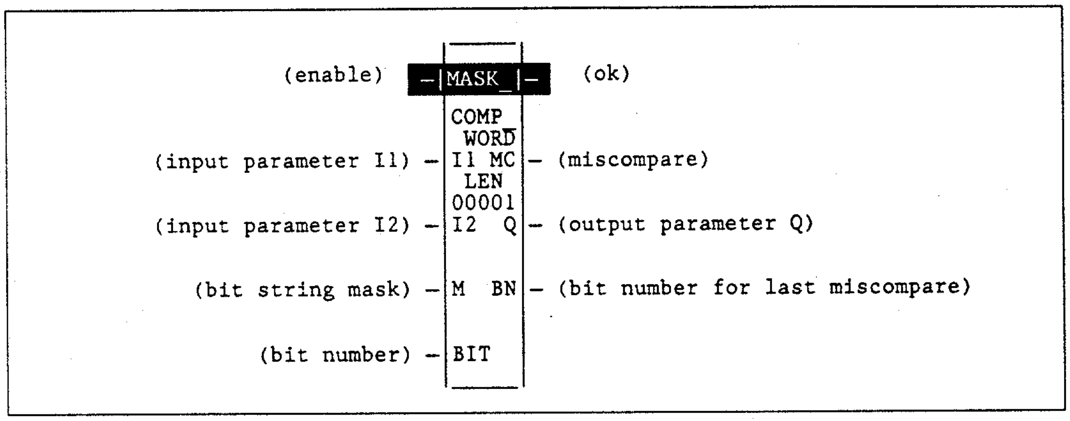
If all corresponding bits in strings I1 and I2 match, the function sets the "miscompare" output MC to 0 and BN to the highest bit number in the input strings. The comparison then stops. On the next invocation of MCMP, it will be reset to 1.

If a Miscompare is Found

When the two bits currently being compared are not the same, the function then checks the correspondingly-numbered bit in string M (the mask). If the mask bit is a 1, the comparison continues until another miscompare or the end of the input strings is reached.

If a miscompare is detected and the corresponding mask bit is a 0, the function:

1. Sets the corresponding mask bit to a 1.
2. Sets the miscompare (MC) output to 1.
3. Updates the output bit string Q to match the new content of mask string M.
4. Sets the bit number output (BN) to the number of the miscompared bit (for example, 6).
5. Stops the comparison.



Parameters:

Parameter	Description
enable	Permissive logic to enable the function.
I1	Reference for the first bit string to be compared.
I2	Reference for the second bit string to be compared.
M	Reference for the bit string mask.
BIT	Reference for the bit number where the next comparison should start.
ok	The ok output is energized when enable is energized.
MC	User logic to determine if a miscompare has occurred.
Q	Output copy of the mask (M) bit string.
BN	Number of the bit where the latest miscompare occurred.
LEN	LEN is the number of words in the bit string.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
I1	•	o	o	o	o	o	o		•	•	•	•	•	•		
I2	•	o	o	o	o	o	o		•	•	•	•	•	•		
M		o	o	o	o	o†	o		•	•	•	•	•	•		
BIT	•	•	•	•	•		•		•	•	•	•	•	•	•	
ok	•															•
MC	•															•
Q		o	o	o	o	o†	o		•	•	•	•	•	•		
BN		•	•	•	•		•		•	•	•	•	•	•		

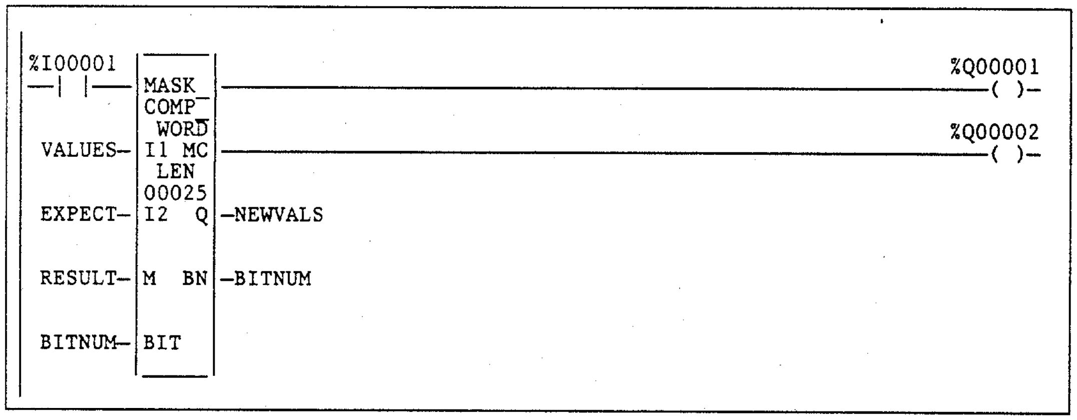
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for WORD data only; not valid for DWORD.
† %SA, %SB, %SC only; %S cannot be used.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, whenever %I00001 is set, the function compares the bits represented by the reference VALUES against the bits represented by the reference EXPECT. Comparison begins at the bit number specified in BITNUM. If an unmasked miscompare is detected, the comparison stops. The corresponding bit is set in the mask RESULT. In addition, the output string NEWVALS is updated with the new value of RESULT, and coil %Q00002 is turned on. Coil %Q00001 is turned on whenever the function receives power flow.



Section 6: Data Move Functions

Data move functions provide basic data move capabilities. This section describes the following data move functions:

Abbreviation	Function	Description	Page
MOVE	Move	Copy data as individual bits. The maximum length allowed is 32,767, except for MOVE_BIT which is 256 bits. Data can be moved into a different data type without prior conversion.	4-76
BLKMOV	Block Move	Copy a block of seven constants to a specified memory location. The constants are input as part of the function.	4-78
BLKCLR	Block Clear	Replace the content of a block of data with all zeros. This function can be used to clear an area of bit (%I, %Q, %M, %G, or %T) or word (%R, %P, %L, %AI, or %AQ) memory. The maximum length allowed is 256 words.	4-80
SHFR	Shift Register	Shift one or more data words into a table. The maximum length allowed is 256 words.	4-82
BITSEQ	Bit Sequencer	Perform a bit sequence shift through an array of bits. The maximum length allowed is 256 words.	4-85
SWAP	Swap	Swap two bytes of data within a word, or two words within a double word. The maximum length allowed is 256 words.	4-90
COMMREQ	Communications Request	Allow the program to communicate with an intelligent module, such as a Bus Controller, Programmable Coprocessor Module, or Subnet Module.	4-92
VMERD	VME Read	Read data from the VME backplane. The maximum length allowed is 32,767.	4-100
VMEWRT	VME Write	Write data to the VME backplane. The maximum length allowed is 32,767.	4-102
VMERMW	VME Read/ Modify/Write	Update a data element using the read/modify/write cycle on the VME bus.	4-104
VMETST	VME Test and Set	Handle semaphores on the VME bus.	4-106
VME_CFG_RD	VME Configuration Read	Read the configuration for a VME module.	4-109
VME_CFG_WRT	VME Configuration Write	Write the configuration to a VME module.	4-112
DATA_INIT	Data Initialization	Copy a block of constant data to a reference range.	4-115
DATA_INIT_COMM	Data Initialize Communications Request	Initialize a COMMREQ function with a block of constant data. The length should equal the size of the COMMREQ function's entire command block.	4-118
DATA_INIT_ASCII	Data Initialize ASCII	Copy a block of constant ASCII text to a reference range. The length must be an even number.	4-121

MOVE (INT, UINT, DINT, BIT, WORD, DWORD, REAL)

Use the MOVE function to copy data (as individual bits) from one location to another. Because the data is copied in bit format, the new location does not need to be the same data type as the original location.

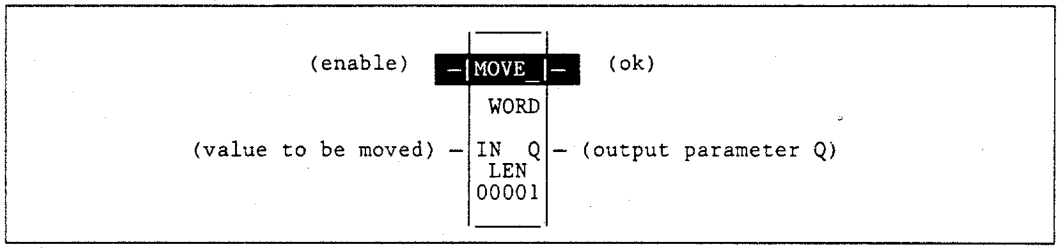
The MOVE function has two input parameters and two output parameters. When the function receives power flow, it copies data from the input parameter IN to the output parameter Q as bits. If data is moved from one location in discrete memory to another, (for example, from %I memory to %T memory), the transition information associated with the discrete memory elements is also copied to the new location. Data at the input parameter does not change unless there is an overlap in the source destination.

Input IN can be either a reference for the data to be moved or a constant. If a constant is specified, then the constant value is placed in the location specified by the output reference. For example, if a constant value of 4 is specified for IN, then 4 is placed in the memory location specified by Q. If the length is greater than 1 and a constant is specified, then the constant is placed in the memory location specified by Q and the locations following, up to the length specified. For example, if the constant value 9 is specified for IN and the length is 4, then 9 is placed in the memory location specified by Q and the three locations following.

The LEN operand specifies the number of:

- Words to be moved for MOVE_INT, MOVE_UINT, and MOVE_WORD.
- Double words to be moved for MOVE_DINT and MOVE_DWORD.
- Bits to be moved for MOVE_BIT.
- Reals to be moved for MOVE_REAL.

The function passes power to the right whenever power is received.



Parameters:

Parameter	Description
enable	When the function is enabled, the move is performed.
IN	IN contains the value to be moved. For MOVE_BIT, any discrete reference may be used; it does not need to be byte aligned. However, 1 bit, beginning with the reference address specified, is displayed online.
ok	The ok output is energized whenever the function is enabled.
Q	When the move is performed, the value at IN is written to Q. For MOVE_BIT, any discrete reference may be used; it does not need to be byte aligned. However, 1 bit, beginning with the reference address specified, is displayed online.
LEN	LEN must be between 1 and 32,767, except for MOVE_BIT, which is between 1 and 256 bits, unless IN is a constant. For MOVE_BIT, when IN is a constant, LEN must be between 1 and 16.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
IN	•	o	o	o	o	Δ	o	o	•	•	•	•	•	•	•	
ok	•															•
Q	•	o	o	o	o	†	o	o	•	•	•	•	•	•		

Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).

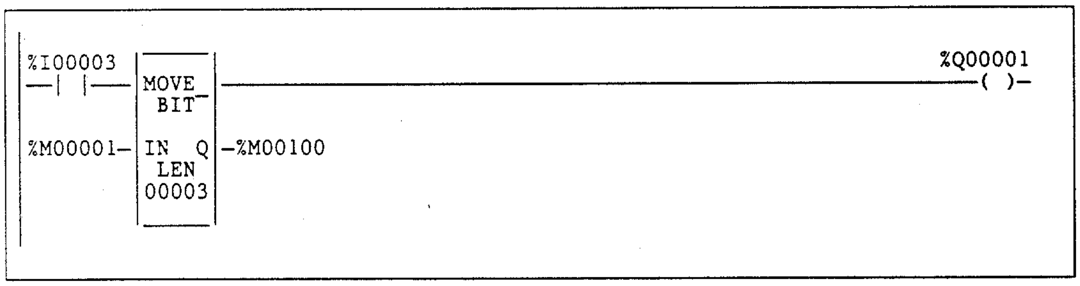
•	Valid reference or place where power may flow through the function.
o	Valid reference for INT, UINT, BIT, or WORD data only; not valid for DINT, DWORD, or REAL. For MOVE_BIT, discrete user references %I, %Q, %M, and %T need not be byte aligned. %U is allowed for MOVE_BIT only.
Δ	Valid reference for BIT or WORD data only; not valid for INT, UINT, DINT, DWORD, or REAL.
†	%SA, %SB, %SC only; %S cannot be used.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In this example, whenever %I00003 is set, the three bits %M00001, %M00002, and %M00003 are moved to %M00100, %M00101, and %M00102, respectively. Coil %Q00001 is turned on.



BLKMOV (INT, UINT, DINT, WORD, DWORD, REAL)

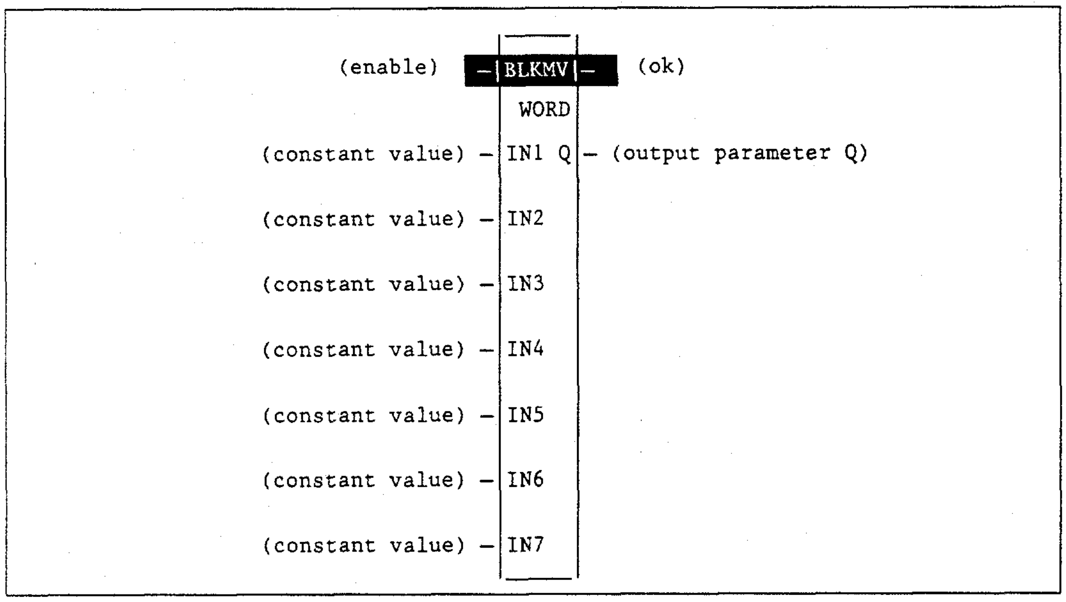
Use the Block Move (BLKMOV) function to copy a block of seven constants to a specified location.

The BLKMOV function has eight input parameters and two output parameters. When the function receives power flow, it copies the constant values into consecutive locations, beginning at the destination specified in output Q. Output Q cannot be the input of another program function.

Note

For BLKMOV_INT, the values of IN1 — IN 7 are displayed as signed decimals. For BLKMOV_WORD, IN1 — IN7 are displayed in hexadecimal.

The function passes power to the right whenever power is received.



Parameters:

Parameter	Description
enable	When the function is enabled the block move is performed.
IN1 - IN7	IN1 through IN7 contain seven constant values.
ok	The ok output is energized whenever the function is enabled.
Q	Output Q contains the first integer of the moved array. IN1 is moved to Q.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
IN1-IN7															•	
ok	•															•
Q		o	o	o	o	Δ†	o		•	•	•	•	•	•		

Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).

• Valid reference or place where power may flow through the function.

o Valid reference for INT, UINT, or WORD only; not valid for DINT, DWORD, or REAL.

Δ Valid reference for WORD data only; not valid for INT, UINT, DINT, DWORD, or REAL.

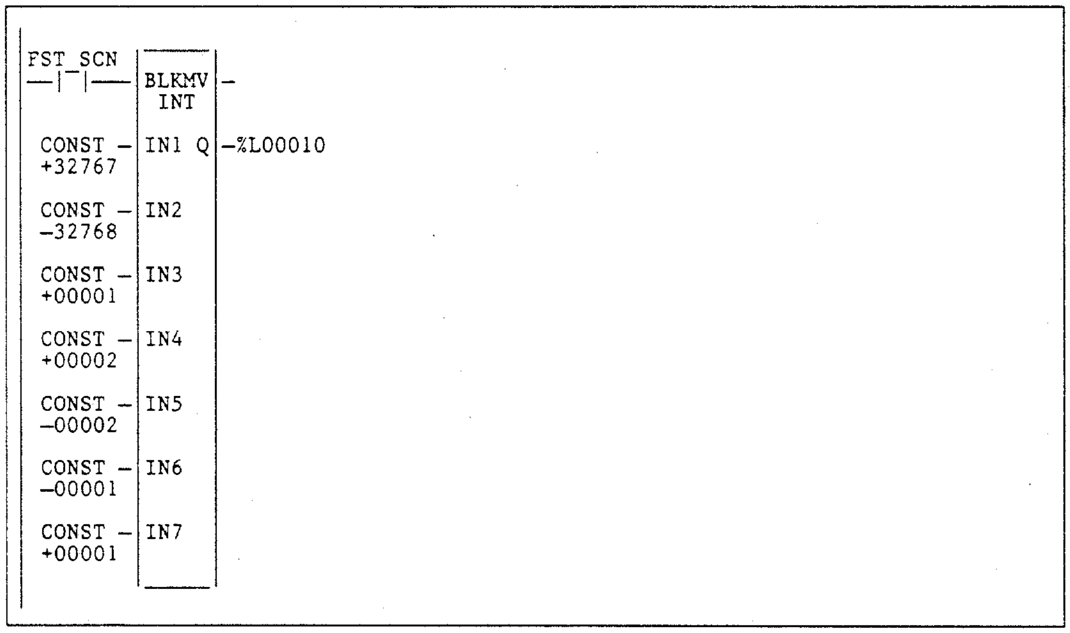
† %SA, %SB, %SC only; %S cannot be used.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, when the enabling input represented by the nickname FST_SCN is ON, the BLKMOV function copies the seven input constants into memory locations %L00010 through %L00016.



BLKCLR (WORD)

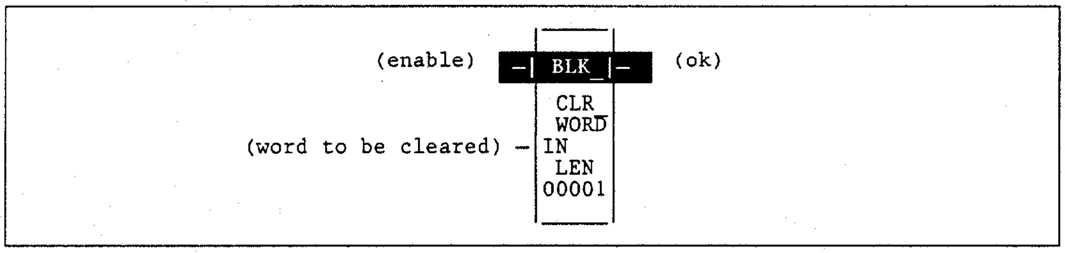
Use the Block Clear (BLKCLR) function to fill a specified block of data with zeros.

The BLKCLR function has two input parameters and one output parameter. When the function receives power flow, it writes zeros into the memory location beginning at the reference specified by IN. When the data to be cleared is from discrete memory (%I, %Q, %M, %G, or %T), the transition information associated with the references is also cleared.

Note

The input parameter IN is not included in coil checking.

The function passes power to the right whenever power is received.



Parameters:

Parameter	Description
enable	When the function is enabled, the array is cleared.
IN	IN contains the first word of the array to be cleared.
ok	The ok output is energized whenever the function is enabled.
LEN	LEN must be between 1 and 256 words.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
Q		•	•	•	•	•†	•		•	•	•	•	•	•		
ok	•															•

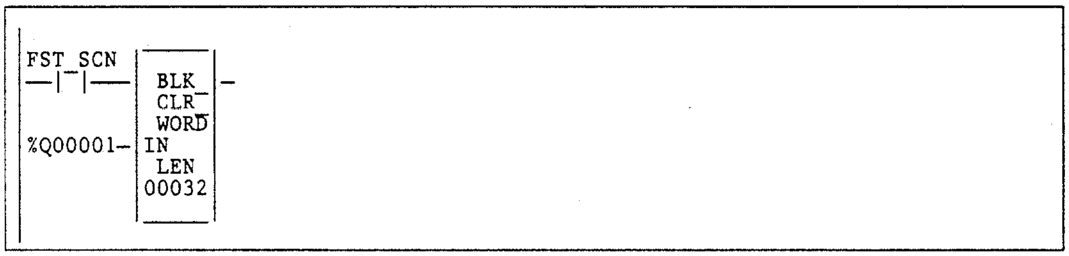
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
† %SA, %SB, %SC only; %S cannot be used.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, at power-up, 32 words of %Q memory (512 points) beginning at %Q00001 are filled with zeros. The transition information associated with these references will also be cleared.



SHFR (BIT, WORD, DWORD)

Use the Shift Register (SHFR) function to shift one or more data words or data bits from a reference location into a specified area of memory. For example, one word might be shifted into an area of memory with a specified length of five words. As a result of this shift, another word of data would be shifted out of the end of the memory area.

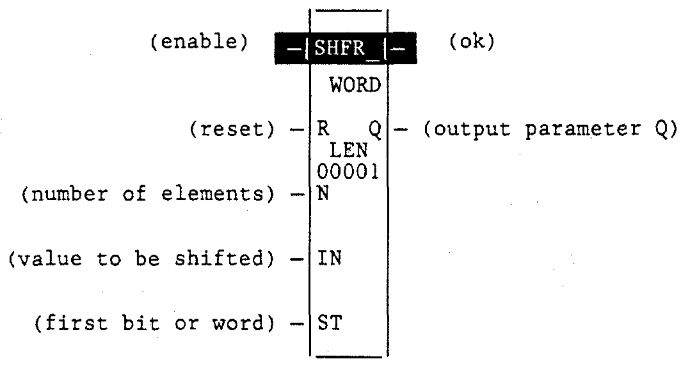
Note

When assigning reference addresses, overlapping input and output reference address ranges in multi-word functions may produce unexpected results.

The SHFR function has five input parameters and two output parameters. The reset input (R) takes precedence over the function enable input. When the reset is active, all references beginning at the shift register (ST) up to the length specified for LEN, are filled with zeros. LEN determines the length of the shift register.

If the function receives power flow and reset is not active, it shifts data in the shift register down by the number of elements (bit or word) specified in N. The last element in the shift register is shifted into Q. The rightmost element of IN is shifted into the vacated element starting at ST. The contents of the shift register are accessible throughout the program because they are overlaid on absolute locations in logic addressable memory.

The function passes power to the right whenever power is received through the enable logic.



Parameters:

Parameter	Description
enable	When enable is energized and R is not, the shift is performed.
R	When R is energized, the shift register located at ST is filled with zeros.
N	N contains the number of elements to be shifted into the shift register.
IN	IN contains the value to be shifted into the first bit or word of the shift register. For SHFR_BIT, any discrete reference may be used; it does not need to be byte aligned. However, 1 bit, beginning with the reference address specified, is displayed online.
ST	ST contains the first bit or word of the shift register. For SHFR_BIT, any discrete reference may be used; it does not need to be byte aligned. However, 16 bits, beginning with the reference address specified, are displayed online.
ok	The ok output is energized whenever the function is enabled.
Q	Output Q contains the bit or word shifted out of the shift register. For SHFR_BIT, any discrete reference may be used; it does not need to be byte aligned. However, 1 bit, beginning with the reference address specified, is displayed online.
LEN	LEN must be between 1 and 256 bits, words, or double words.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
R	•															
N															•	
IN	*	o	o	o	o	o	o		•	•	•	•	•	•	•	
ST		o	o	o	o	o†	o		•	•	•	•	•	•		
ok	•															•
Q	*	o	o	o	o	o†	o	o	•	•	•	•	•	•		

Note: *Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).*

•	Valid reference or place where power may flow through the function.
*	Valid reference for WORD or DWORD data only; not valid for BIT.
o	Valid reference for BIT or WORD data only; not valid for DWORD. For SHFR_BIT, discrete user references %I, %Q, %M, and %T need not be byte aligned.
†	%SA, %SB, %SC only; %S cannot be used.

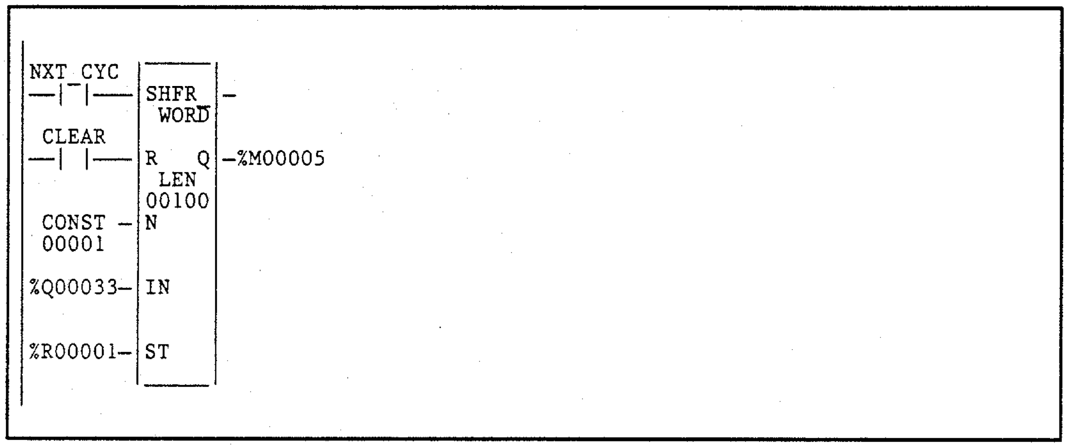
Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example 1:

In the following example, the shift register operates on register memory locations %R00001 through %R00100. When the reset reference CLEAR is active, the shift register words are set to zero.

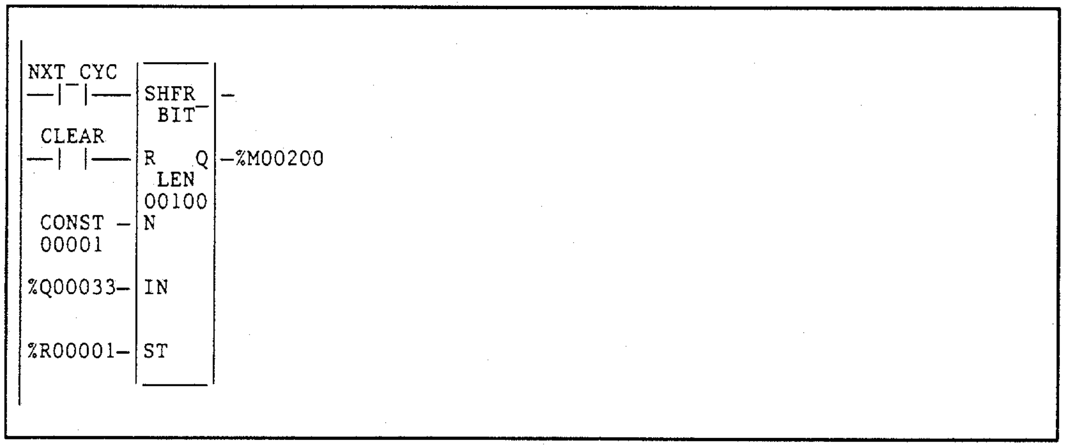
When the NXT_CYC reference is active and CLEAR is not active, the word from output status table location %Q00033 is shifted into the shift register. The word shifted out of the shift register is stored in output %M00005. Note that, for this example, the length specified for LEN and the amount of data to be shifted (N) are not the same.



Example 2:

In this example, the shift register operates on memory locations %M00001 through %M00100. When the reset reference CLEAR is active, the SHFR function fills %M00001 through %M00100 with zeros.

When NXT_CYC is active and CLEAR is not, the SHFR function shifts the data in %M00001 to %M00100 down by one bit. The bit in %Q00033 is shifted into %M00001 while the bit shifted out of %M00100 is written to %M00200.



BITSEQ (BIT)

The Bit Sequencer (BITSEQ) function performs a bit sequence shift through an array of bits. The BITSEQ function has five input parameters and one output parameter. The operation of the function depends on the previous value of the parameter EN, as shown in the following table.

R Current Execution	EN Previous Execution	EN Current Execution	Bit Sequencer Execution
OFF	OFF	OFF	Bit sequencer does not execute.
OFF	OFF	ON	Bit sequencer increments/decrements by 1.
OFF	ON	OFF	Bit sequencer does not execute.
OFF	ON	ON	Bit sequencer does not execute.
ON	ON/OFF	ON/OFF	Bit sequencer resets.

The reset input (R) overrides the enable (EN) and always resets the sequencer. When R is active, the current step number is set to the value passed in via the step number parameter. If no step number is passed in, step is set to 1. All of the bits in the sequencer are set to 0, except for the bit pointed to by the current step, which is set to 1.

When EN is active and R is not active, the bit pointed to by the current step number is cleared. The current step number is either incremented or decremented, based on the direction parameter. Then, the bit pointed to by the new step number is set to 1.

- When the step number is being incremented and it goes outside the range of (1 ≤ step number ≥ LEN), it is set back to 1.
- When the step number is being decremented and it goes outside the range of (1 ≤ step number ≥ LEN), it is set to LEN.

The parameter ST is optional. If it is not used, the BITSEQ operates as described above, except that no bits are set or cleared. Basically, the BITSEQ then just cycles the current step number through its legal range.

Memory Required for a Bit Sequencer

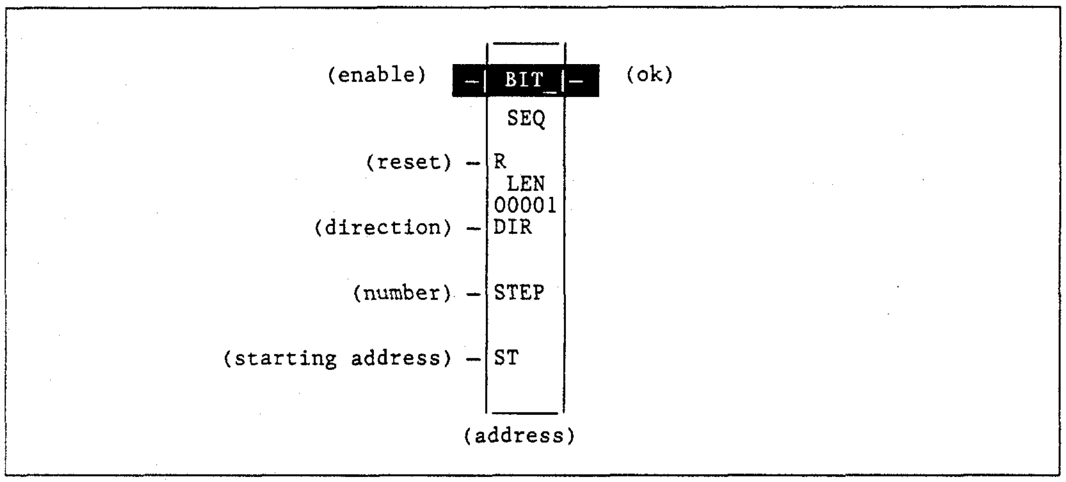
Each bit sequencer uses three words (registers) of %R, %L, or %P memory to store the following information:

current step number	word 1
length of sequence (in bits)	word 2
control word	word 3

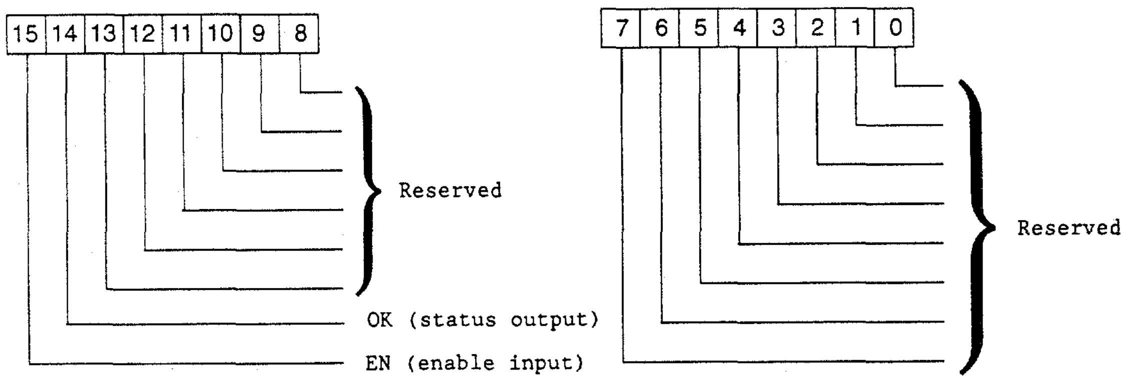
Note

Do **not** write to these registers from other functions.

When you enter a bit sequencer, you must enter an address for the location of these three consecutive words (registers) directly below the graphic representing the function. For example:



The control word stores the state of the boolean inputs and outputs of its associated function block, as shown in the following format:



Note

Bits 0 through 13 are not used.

Parameters:

Parameter	Description
address	The bit sequencer uses three consecutive words (registers) of %R, %P, or %L memory to store the: • Current step number = word 1. • Length of sequence in bits = word 2. • Control word = word 3. When you enter a bit sequencer, you must enter an address for the location of these three consecutive words (registers) directly below the graphic representing the function. For more information, refer to the preceding page. Note: Do not use this address with other instructions. Caution: Overlapping references will result in erratic operation of the bit sequencer.
enable	When the function is enabled, if it was not enabled on the previous sweep and if R is not energized, the bit sequence shift is performed.
R	When R is energized, the bit sequencer's step number is set to the value in STEP (default = 1), and the bit sequencer is filled with zeros, except for the current step number bit.
DIR	When DIR is energized, the bit sequencer's step number is incremented prior to the shift. Otherwise, it is decremented.
STEP	When R is energized, the step number is set to this value.
ST	ST contains the first word of the bit sequencer.
ok	The ok output is energized whenever the function is enabled.
LEN	LEN must be between 1 and 256 words.

Note

Coil checking for the BITSEQ function checks for 16 bits from the ST parameter, even when LEN is less than 16.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
address									•					•		
enable	•															
R	•															
DIR	•															
STEP		•	•	•	•		•		•	•	•	•	•	•	•	•
ST		•	•	•	•	†	•		•	•	•		•	•		•
ok	•															

• Valid reference or place where power may flow through the function.
† %SA, %SB, %SC only; %S cannot be used.

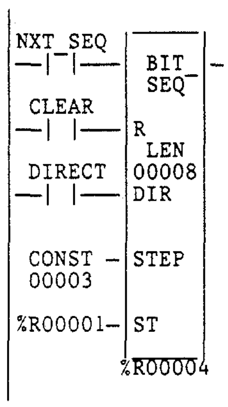
Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, the sequencer operates on register memory %R00001. Its static data is stored in registers R00004, R00005, and R00006. When CLEAR is active, the sequencer is reset and the current step is set to step number 3. The first 8 bits of %R00001 are set to zero.

When NXT_SEQ is active and CLEAR is not active, the bit for step number 3 is cleared and the bit for step number 2 or 4 (depending on whether DIR is energized) is set.



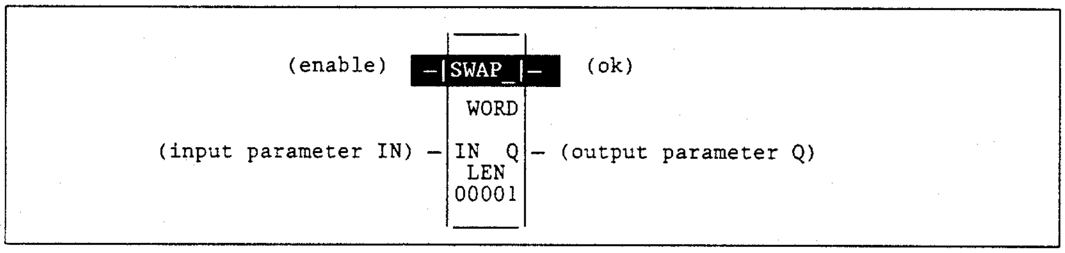
SWAP (WORD, DWORD)

Use the SWAP function to swap two bytes within a word, or two words within a double word.

The SWAP function can be performed over a range of memory by specifying a length greater than 1 for the function. If this is done, each word or double word of data within the specified length is appropriately swapped.

The SWAP function has two input parameters and two output parameters. When the SWAP function receives power flow, it performs the swap operation on each word or double word of data within the specified area. The results of the swap are stored to output Q.

The SWAP function passes power to the right whenever power is received.



Parameters:

Parameter	Description
enable	When the function is enabled, the swap is performed.
IN	IN contains the data to be swapped.
ok	The ok output is energized when enable is energized.
Q	Output Q contains the swapped version of the original data IN.
LEN	LEN must be between 1 and 256 words.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
IN	•	o	o	o	o		o		•	•	•	•	•	•	•	
ok	•															•
Q	•	o	o	o	o		o		•	•	•	•	•	•		

Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).

• Valid reference or place where power may flow through the function.

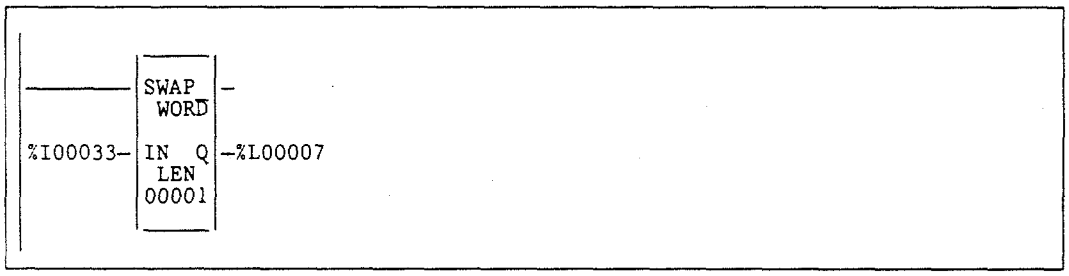
o Valid reference for WORD data only; not valid for DWORD.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, two bytes located in bits %I00033 through %I00048 are swapped. The result is stored in %L00007.



COMMREQ

Use the Communication Request (COMMREQ) function if the program needs to communicate with an intelligent module such as a Bus Controller, Programmable Coprocessor Module, or LAN Interface Module. The information presented on the following pages shows the format of the COMMREQ function. You will need additional information to program the COMMREQ for each type of device. Programming requirements for each type of module that uses the COMMREQ function are described in the module's documentation.

Note

If you are using Serial Communications, refer to the *Series 90™ PLC Serial Communications User's Manual* (GFK-0582). If you are using MMS-Ethernet Communications, refer to the *MMS-Ethernet Communications for the Series 90™-70 PLC User's Manual* (GFK-0686).

As an example of the types of communications that can be requested, the Genius Bus Controller uses the following types of messages:

Command	Description
1	Pulse test outputs.
2	Read configuration data from a device on the bus.
3	Write configuration data to a device on the bus.
4	Read diagnostics.
5	Clear circuit fault.
6	Clear all circuit faults.
7	Assign monitor.
8	Enable/disable outputs.
9	Enable/disable global data.
10	Switch Bus Switching Module.
11	Read device.
12	Write device.
13	Dequeue datagram.
14	Read datagram.
15	Receive datagram.

The Send Datagram and Receive Datagram commands (14 and 15 above) can be used to transmit several additional messages. Refer to the *Bus Controller Manual,* GFK-0398, for programming information on the bus controller.

Note that devices can also exchange global data with the PLC during the System Communications Window, which occurs automatically at the end of each sweep. Such communications do not require program assistance.

The COMMREQ function has four input parameters and two output parameters. When the COMMREQ function receives power flow, a command block of data is sent to the communications TASK. The command block begins at the reference specified using the parameter IN. The device to be communicated with is indicated by entering its rack and slot number for SYSID.

The COMMREQ may either send a message and wait for a reply, or send a message and continue without waiting for a reply. If the command block specifies that the program will not wait for a reply, the command block contents are sent to the receiving device and the program execution resumes immediately. (The timeout value is ignored.) This is referred to as **NOWAIT** mode.

If the command block specifies that the program will wait for a reply, the command block contents are sent to the receiving device and the CPU waits for a reply. The maximum length of time the PLC will wait for the device to respond is specified in the command block. If the device does not respond in that time, program execution resumes. This is referred to as **WAIT** mode.

The function passes power flow, unless the timeout period is exceeded, or if a 0 timeout period has been specified. The Function Faulted (FT) output may be set ON if:

1. The specified target module is not present or is faulted.
2. The specified task is not valid for the device.
3. The data length is 0.

The Function Faulted output may have these states:

Enable	Error?	Function Faulted Output
active	no	OFF
active	yes	ON
not active	no execution	OFF

Command Block

The command block provides additional information needed by the COMMREQ function.

The address of the command block is specified for the IN input to the COMMREQ function. This address may be in any word-oriented user reference (%R, %L, %P, %AI, or %AQ). The length of the command block depends on the specific command being sent.

The command block has the following structure:

Data Block Length	address
Wait/No Wait Flag	address + 1
Status Pointer Memory Type	address + 2
Status Pointer Offset	address + 3
Idle Timeout Value	address + 4
Maximum Communication Time	address + 5
Data Block	address + 6 to address + 133

Information required for the command block can be placed in the designated memory area using the MOV, BLKMOV, or DATA_INIT function block.

When entering information for the command block, refer to these definitions:

Data Block Length: The number of data words starting with the data at address + 6 to the end of the command block, inclusive. The data block length ranges from 1 to 128 words. Each COMMREQ command has its own data block length.

Wait/No Wait Flag: This selects whether or not the program should wait for CCM communications to be completed.

For	Enter
No wait	0
Wait for reply	1

The flag bit is stored in the least significant bit (LSB) at address + 1. The rest of the word should be filled with zeros.

Status Pointer Memory Type: The two status pointer words specify a PLC memory location where the status word returned by the device will be written when the COMMREQ completes.

Status Pointer Memory Type	address + 2
Status Pointer Offset	address + 3

Status pointer memory type contains a numeric code that specifies the user reference memory type for the status word. The table below shows the code for each reference type:

For This Memory Type		Enter This Value *
%I	Discrete input table (**BIT** mode)	70
%Q	Discrete output table (**BIT** mode)	72
%I	Discrete input table (**BYTE** mode)	16
%Q	Discrete output table (**BYTE** mode)	18
%R	Register memory	8
%AI	Analog input table	10
%AQ	Analog output table	12

* Numbers shown are decimal.

Note

The value entered determines the mode. For example, if you enter the %I bit mode is 70, then the offset will be viewed as that bit. On the other hand, if the %I value is 16, then the offset will be viewed as that byte.

The high byte at address + 2 should contain zero.

Status Pointer Offset: The word at address + 3 contains the offset for the status word within the selected memory type.

Note

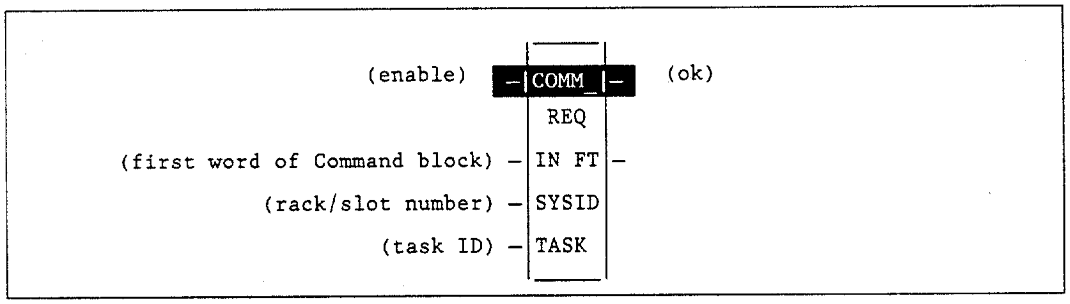
The status pointer offset is a zero-based value. %R00001, for example, is at offset zero in the register table.

Idle Timeout Value: The idle timeout value is the maximum time the PLC CPU waits for the device to acknowledge receipt of the COMMREQ. This value is ignored in **NOWAIT** mode. If **WAIT** mode is selected, address + 4 specifies the idle timeout period in 100-microsecond increments.

Maximum Communication Time: The value at address +5 specifies the maximum time the PLC CPU waits for the device to complete the COMMREQ. This time is also specified in 100-microsecond increments and is ignored in **NOWAIT** mode.

Data Block

The data block contains the parameters of the command. The data block begins with a command number in address + 6, which identifies the type of communications function to be performed. Refer to the specific device manual (i.e., PCM, GBC, Communications) for specific COMMREQ command formats.



Parameters:

Parameter	Description
enable	When the function is energized, the communications request is performed.
IN	IN contains the first word of the command block.
SYSID	SYSID contains the rack number (most significant byte) and slot number (least significant byte) of the target device.
TASK	TASK contains the task ID of the process on the target device.
FT	FT is energized when the communication request fails. This was previously discussed in greater detail on page 4-93.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
IN									•	•	•	•	•	•		
SYSID		•	•	•	•		•		•	•	•	•	•	•	•	
TASK									•	•	•	•	•	•	•	
ok	•															•
FT	•															•

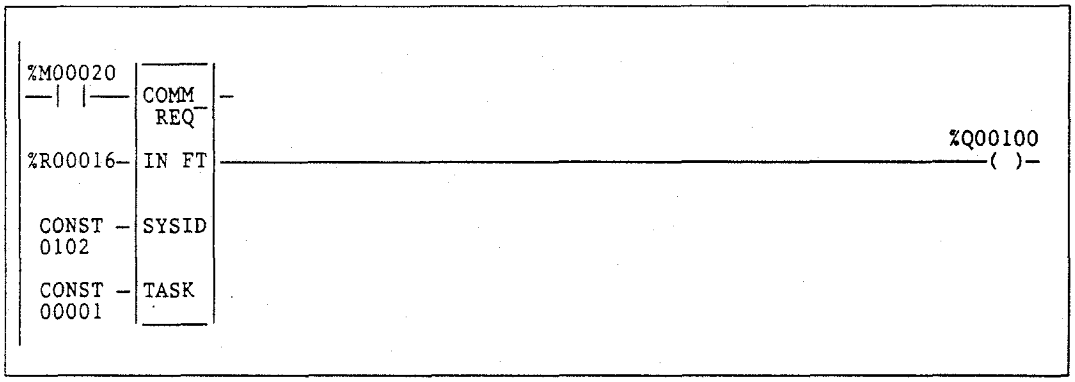
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

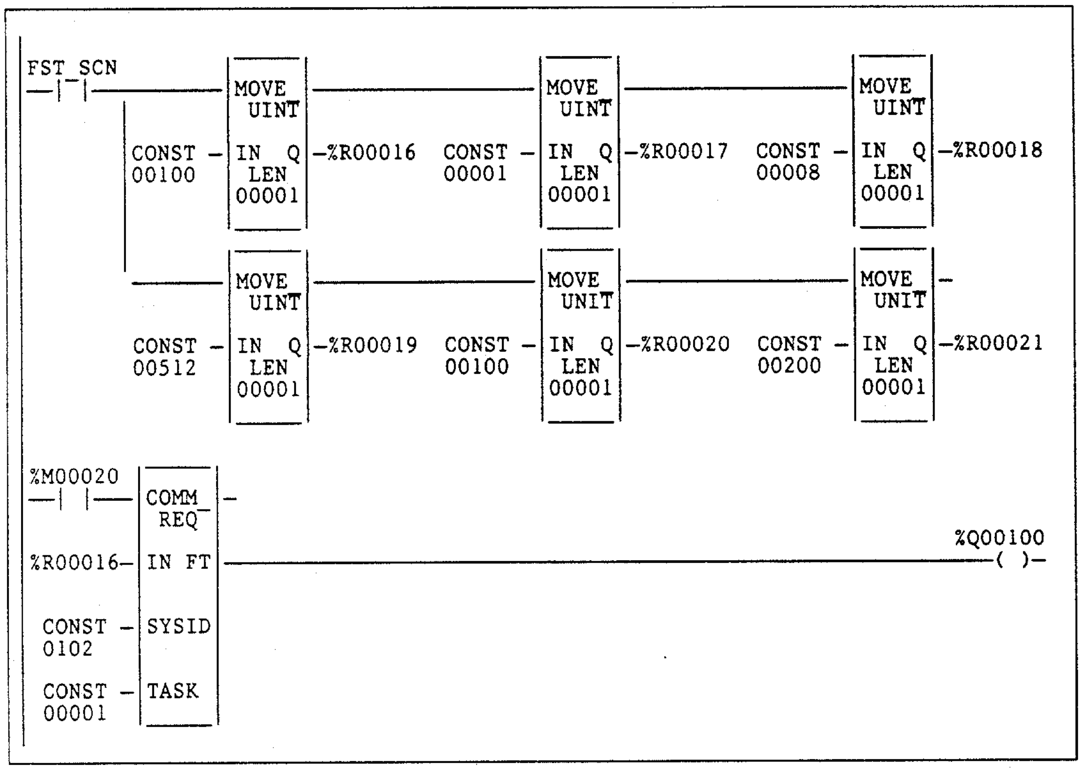
Example 1:

In the following example, when enabling input %M00020 is ON, a command block located starting at %R00016 is sent to communications task 1 in the device located at rack 1, slot 2 of the PLC. If an error occurs, %Q00100 is set.



Example 2:

This example shows how the MOVE function can be used to enter command block contents for the COMMREQ described in example 1:



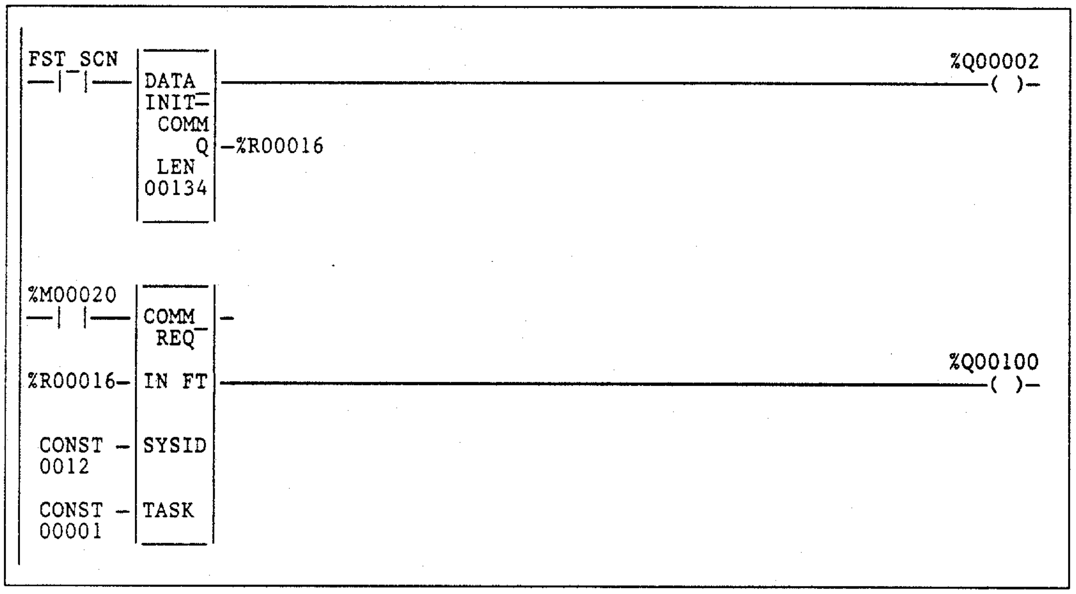
Input IN of the COMMREQ specifies %R00016 as the beginning reference for the command block. Successive references contain the following:

Data Block Length	%R00016
Wait/No Wait Flag	%R00017
Status Pointer Memory Type	%R00018
Status Pointer Offset	%R00019
Idle Timeout Value	%R00020
Maximum Communication Time	%R00021
Data Block	%R00022 to end of data

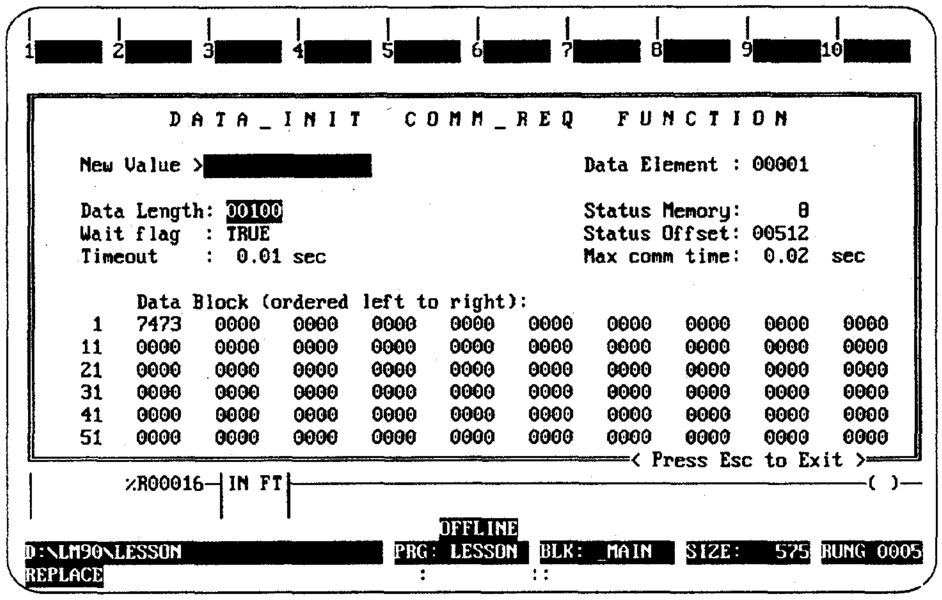
MOVE functions supply the following command block data for the COMMREQ. The first MOVE function places the length of the data being communicated in %R00016. The second MOVE function places the constant 1 in %R00017. This specifies **WAIT** mode.

The third MOVE function places the constant 8 in %R00018. This specifies the register table as the location for the status pointer. The next MOVE function places the constant 512 in reference %R00019. Therefore, the status pointer is located at %R00513. Additional MOVE functions place the constant 100 in %R00020 and 200 in %R00021. 100 is an idle timeout value of 10 milliseconds for the COMMREQ; 200 represents the maximum communications time of 20 milliseconds.

The programming logic displayed in example 2 on the previous page can be simplified by replacing the six MOVE functions with one DATA_INIT_COMM function.



Position the cursor on the DATA_INIT_COMM function block, and press **Zoom (F10)** to zoom into a window similar to the one show below:



```
 1     2     3     4     5     6     7     8     9    10

              D A T A _ I N I T   C O M M _ R E Q   F U N C T I O N

   New Value >                                   Data Element : 00001

   Data Length: 00100                            Status Memory:     8
   Wait flag  : TRUE                             Status Offset: 00512
   Timeout    :  0.01 sec                        Max comm time:  0.02  sec

          Data Block (ordered left to right):
    1     7473  0000  0000  0000  0000  0000  0000  0000  0000  0000
   11     0000  0000  0000  0000  0000  0000  0000  0000  0000  0000
   21     0000  0000  0000  0000  0000  0000  0000  0000  0000  0000
   31     0000  0000  0000  0000  0000  0000  0000  0000  0000  0000
   41     0000  0000  0000  0000  0000  0000  0000  0000  0000  0000
   51     0000  0000  0000  0000  0000  0000  0000  0000  0000  0000
                                                   < Press Esc to Exit >
       %R00016-|IN FT|---------------------------------------------( )-
                                      OFFLINE
D:\LM90\LESSON                        PRG: LESSON  BLK: _MAIN  SIZE:   575  RUNG 0005
REPLACE                                    :          ::
```

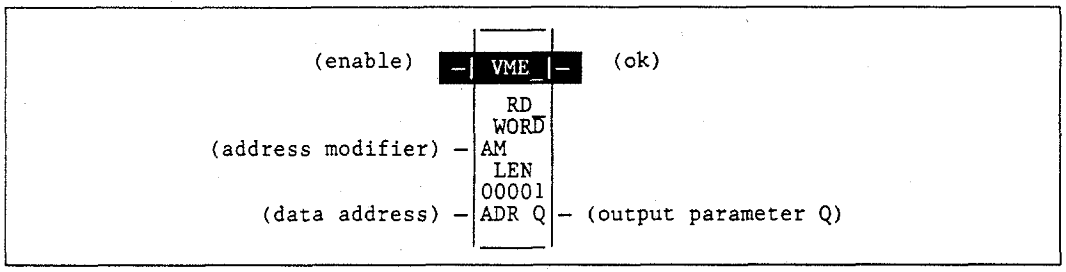
VMERD (BYTE, WORD)

The VME Read (VMERD) function is used to read data from the VME bus.

Note

Using a VME function (VMERD, VMEWRT, VMERMW, or VMETST) requires additional information on the correct way to address the VME board. This information may be obtained from one of two sources. For a qualified VME board, the VME board vendor may issue application notes on the correct use of the board. Otherwise, refer to the *Guidelines for the Selection of Third-Party VME Modules,* GFK-0448.

The VMERD function has three input parameters and one output parameter. When the VMERD function receives power flow, the function accesses the VME module at the address specified by ADR and the address modifier AM. It copies data with the length LEN to PLC locations beginning at output Q. The VMERD function passes power to the right when its operation is successful.



Parameters:

Parameter	Description
enable	When enable is energized, the VME read is performed.
AM	AM contains the address modifier.
ADR	ADR contains the address of the data to be read.
ok	The ok output is energized when the function is enabled and the data is successfully read.
Q	Output Q contains the data read from the address specified by ADR and AM.
LEN	LEN may be 1 to 32,767.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
AM	•								•	•	•	•	•	•	•	
ADR	•								•	•	•	•	•	•	•	
ok	•															•
Q	o	o	o	o	o		o		•	•	•	•	•	•		

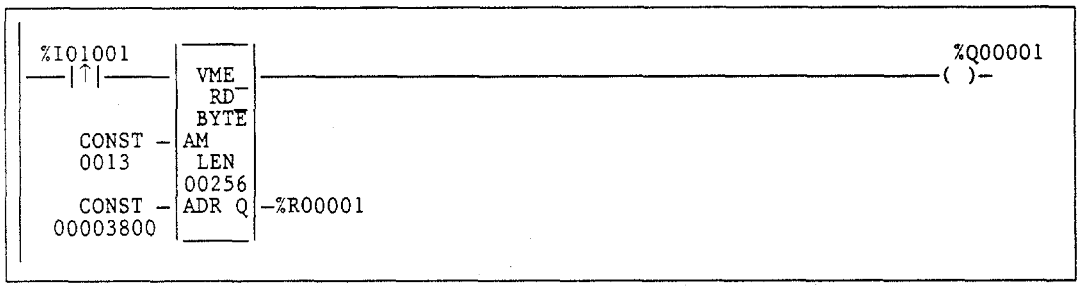
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for BYTE data only; not valid for WORD.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, when enabling input %I01001 goes ON, 256 bytes of short supervisory space are read from address 3800 on the VME bus into registers %R00001 through %R00128. (In a multiple rack system with a BTM, this would be on rack 4.) Unless an error occurs while reading the data, coil %Q00001 is set ON.



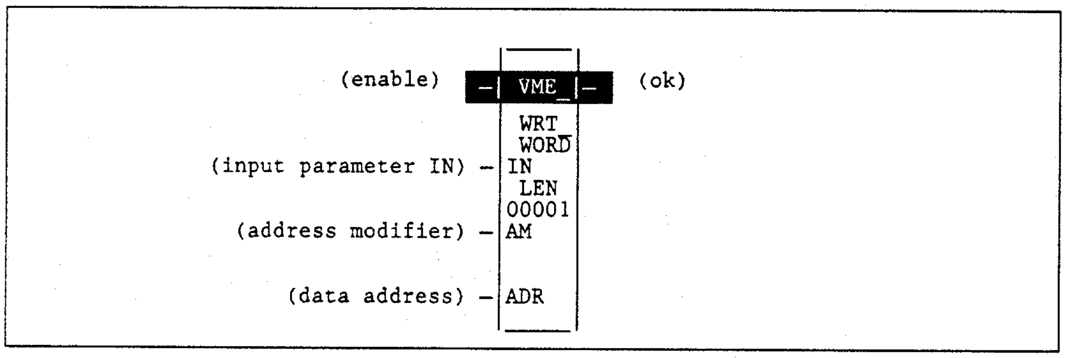
VMEWRT (BYTE, WORD)

The VME Write (VMEWRT) function is used to write data to the VME bus.

Note

Using a VME function (VMERD, VMEWRT, VMERMW, or VMETST) requires additional information on the correct way to address the VME board. This information may be obtained from one of two sources. For a qualified VME board, the VME board vendor may issue application notes on the correct use of the board. Otherwise, refer to the *Guidelines for the Selection of Third-Party VME Modules,* GFK-0448.

The VMEWRT function has four input parameters and one output parameter. When the VMEWRT function receives power flow, the function copies the data from the input parameter IN to the VME module at the address specified in ADR and the address modifier AM. The VMEWRT function passes power to the right to indicate a successful transfer of data.



Parameters:

Parameter	Description
enable	When enable is energized, the VME write is performed.
IN	IN contains the data to be written to the address specified by ADR and AM.
AM	AM contains the address modifier.
ADR	ADR contains the address where the data is to be written.
LEN	LEN may be 1 to 32,767.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
IN	•	o	o	o	o		o		•	•	•	•	•	•	•	
AM	•	•	•	•	•		•		•	•	•	•	•	•	•	
ADR	•								•	•	•	•	•	•	•	
ok	•															•

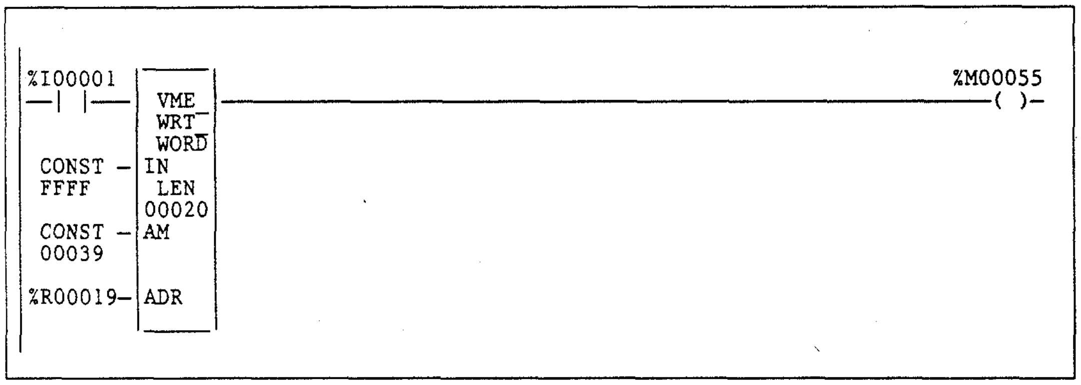
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for BYTE data only; not valid for WORD.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, when enabling input %I00001 is ON, the hexadecimal value FFFF is written to each of 20 words on the VME bus, the first (lowest address) being specified by the content of %R00019 (low word) and %R00020 (high word). Unless an error occurs while writing the data, internal reference %M00055 is set ON.



VMERMW (BYTE, WORD)

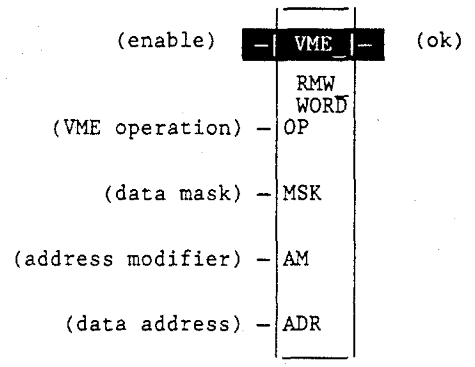
The VME Read/Modify/Write (VMERMW) function is used to update a data element on the VME bus.

Note

Using a VME function (VMERD, VMEWRT, VMERMW, or VMETST) requires additional information on the correct way to address the VME board. This information may be obtained from one of two sources. For a qualified VME board, the VME board vendor may issue application notes on the correct use of the board. Otherwise, refer to the *Guidelines for the Selection of Third-Party VME Modules,* GFK-0448.

The VMERMW function has five input parameters and one output parameter. When the VMERMW function receives power flow, the function reads the data from the board at the address specified by ADR and address modifier AM. This byte or word of data is combined (AND/OR) with the data mask MSK. Selection of AND or OR is made using the OP input. MSK is a word value. If byte data is operated on, only the lower 8 bits of MSK are used.

The result is then written back to the same VME address from which it was read. The VMERMW function passes power flow to the right whenever power is received unless an error occurs.



Parameters:

Parameter	Description
enable	When enable is energized, the VME function is performed.
OP	OP specifies whether data is to be ANDed or ORed with the MSK data.
MSK	MSK contains the data mask.
AM	AM contains the address modifier.
ok	The ok output is energized whenever the function is enabled and performed without error.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
OP															•	
MSK	•	•	•	•	•		•		•	•	•	•	•	•	•	
AM	•	•	•	•	•		•		•	•	•	•	•	•	•	
ADR	•								•	•	•	•	•	•	•	
ok	•															•

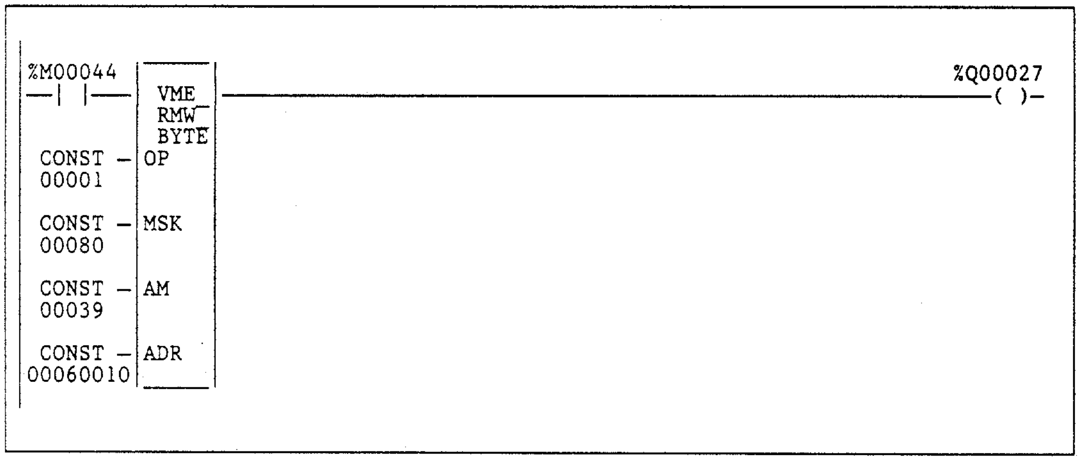
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, when enabling input %M00044 is ON, the hexadecimal value 80H is ORed with the byte of data read from address 060010H on the VME bus in rack 0 (the main rack) using Standard Non-Privileged Data Access. Unless an error occurs while accessing the data, coil %Q00027 is set ON.



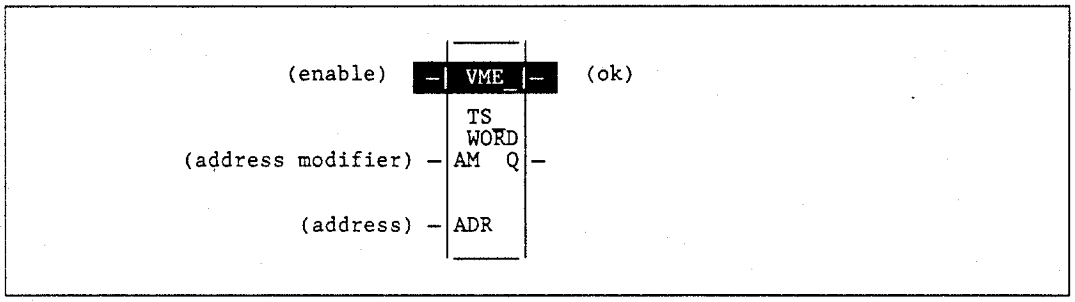
VMETST (BYTE, WORD)

Use the VME Test and Set (VMETST) function to handle semaphores on the VME bus. The VMETST function exchanges a boolean ON (1) for the value currently at the semaphore location. If that value was already ON, then the VMETST function does not obtain the semaphore. If the existing value was OFF, then the semaphore is reset and the VMETST function has the semaphore and the use of the memory area it controls. The semaphore is cleared using the VMEWRT function to write a 0 to the semaphore location.

Note

Using a VME function (VMERD, VMEWRT, VMERMW, or VMETST) requires additional information on the correct way to address the VME board. This information may be obtained from one of two sources. For a qualified VME board, the VME board vendor may issue application notes on the correct use of the board. Otherwise, refer to the *Guidelines for the Selection of Third-Party VME Modules,* GFK-0448.

The VMETST function has three input parameters and two output parameters. When the VMETST function receives power flow, a boolean ON is exchanged with the data at the address specified by ADR using the address modifier specified by AM. The VMETST function sets the Q output to ON if the semaphore was available (OFF) and was acquired. The function passes power flow to the right whenever power is received and no error occurs during execution.



Parameters:

Parameter	Description
enable	When enable is energized, the VME test and set is performed.
AM	AM contains the address modifier.
ADR	ADR contains the address of the semaphore.
ok	The ok output is energized when the function is enabled and performed without error.
Q	Output Q is set ON if the semaphore was available (OFF). Otherwise, Q is set OFF.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
AM	•								•	•	•	•	•	•	•	
ADR	•								•	•	•	•	•	•	•	
ok	•															•
Q	•															•

Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.

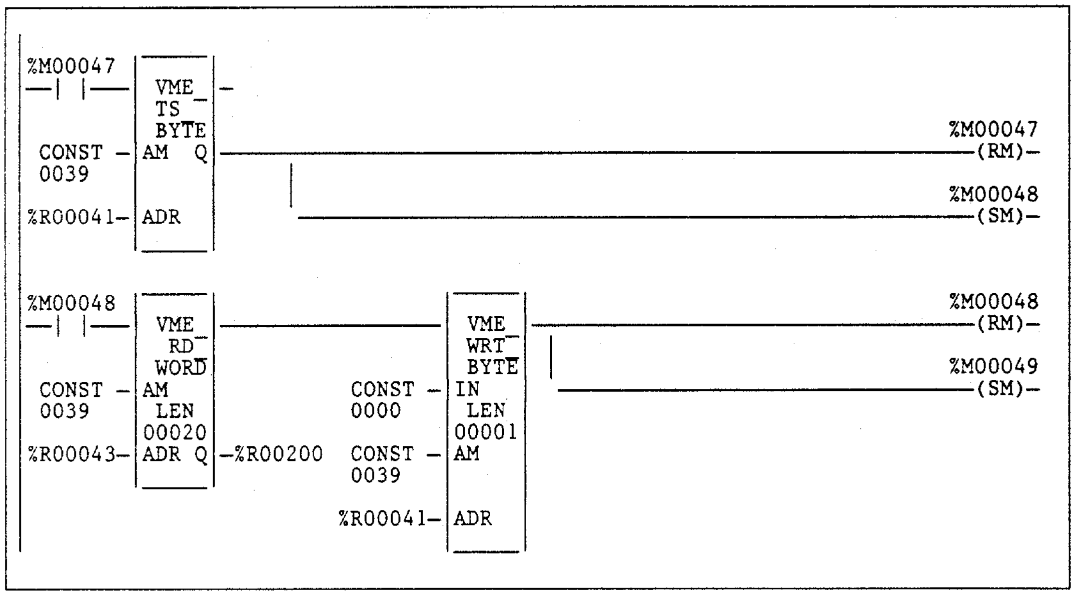
Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, the VMERD, VMEWRT, and VMETST functions are used to read data protected by a semaphore. When enabling input %M00047 is ON, the VMETST function is executed to obtain the semaphore. The semaphore VME address is stored in %R00041 and %R00042 using Standard Non-Privileged Data Access. When this is successful, coil %M00047 is reset and coil %M00048 is set. When %M00048 is set, the VMERD function reads the data (20 words of data whose VME address is stored in %R00043 and %R00044, data read into %R00200 through %R00219). When the read is successful (something is broken or misprogrammed if it is not), the VMTWRT function relinquishes the semaphore. Coil %M00048 is reset when the VMEWRT is successful. %M00049 is set to indicate that fresh data is now available.

If the semaphore is not available, VMERD and VMEWRT are not executed. The effect is that setting %M00047 causes the PLC to check the semaphore each sweep until the semaphore is available. When it becomes available, the semaphore is acquired, the data is read, and the semaphore is relinquished. No further action is taken until %M00047 is set again.



VME_CFG_RD

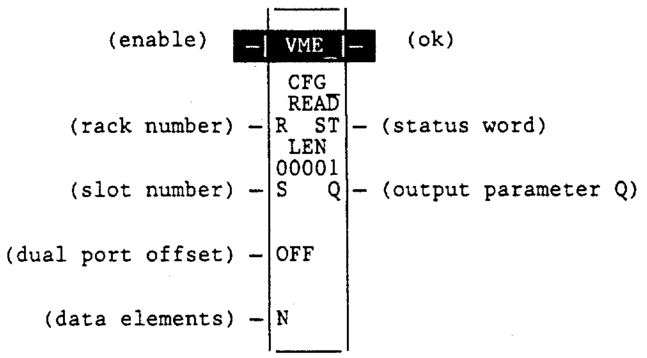
Use the VME_CFG_RD function to read data from the VME bus. The VME_CFG_RD function has five input parameters and three output parameters. When the function receives power, the data elements (N) are read from the VME bus at the location defined by rack (R), slot (S), and dual port offset (OFF). The data read is placed in output Q. The status of the operation is placed in the status word output (ST). The function has a length specification (LEN) of the maximum size of the output array.

Note

The module at the specified rack and slot must be configured as a third-party VME module in BUS INTERFACE mode for this function block to execute successfully. Additional parameters indicating the AM code, the location and size of the module's dual port, and the bus interface type must be specified in the module's configuration for correct operation. See chapter 11 of the *Logicmaster 90-70 Programming Software User's Manual* (GFK-0263) for more information on configuration of third-party VME modules.

If the function is completed successfully, ok is set ON; otherwise, it is set OFF. It is also set OFF when:

- The number of data elements (N) is greater than the length (LEN) specified.
- The rack/slot value (R and S) is out of range or is not a valid VME location.
- The most significant byte of the dual port offset (OFF) is not zero.
- The most significant byte of the dual port address plus the dual port offset is not zero.
- Read beyond the end of dual port memory.
- Specified rack/slot not configured for a Third-Party VME module in **BUS INTERFACE** mode.
- If the dual port offset is an even number, configure for the odd byte only. If the dual port offset is an odd number, configure for word or single word.



Parameters:

Parameter	Description
enable	When the function is enabled, the data initialization is performed.
R	The rack number is specified in R.
S	The slot number is specified in S.
OFF	OFF specifies the dual port offset.
N	N contains the amount of data (data elements) to be read from the VME bus.
ok	The ok output is energized when the function is performed without error.
ST	The status word contains the status of the operation.
Q	When the function is performed, the data is read to array Q.
LEN	LEN is the length of the output array in bytes.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
R	•	•	•	•	•	•	•		•	•	•	•	•	•	•	
S	•	•	•	•	•	•	•		•	•	•	•	•	•	•	
OFF	•					•			•	•	•	•	•	•	•	
N	•	•	•	•	•	•	•		•	•	•	•	•	•	•	
ok	•															•
ST	•	•	•	•	•	†	•		•	•	•	•	•	•		
Q	•	•	•	•	•	†	•		•	•	•	•	•	•		

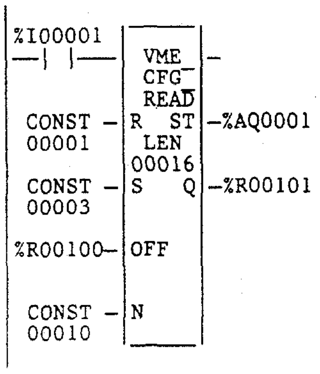
• Valid reference or place where power may flow through the function
† %SA, %SB, %SC only; %S cannot be used.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, when enable is ON, VME data at rack 1, slot 3 and dual port offset defined by %R00100 is read into %R00101 through %R00110 of the array %R00101 through %R00116. If an error was encountered, the status word %AQ0001 will contain an error code.



VME_CFG_WRITE

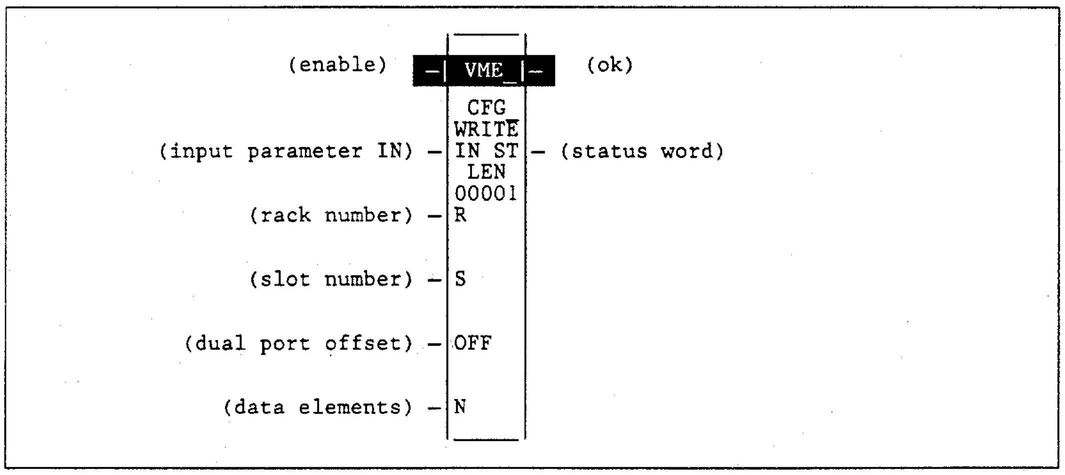
Use the VME_CONFIG_WRITE function to write data from the VME bus. The VME_CFG_WRT function has six input parameters and two output parameters. When the function receives power, the data elements (N) are written from the data array (IN) to the VME bus at the location defined by rack (R), slot (S), and dual port offset (OFF). The status of the operation is placed in the status word output (ST). The function has a length specification (LEN) of the maximum size of the output array.

Note

The module at the specified rack and slot must be configured as a third-party VME module in BUS INTERFACE mode for this function block to execute successfully. Additional parameters indicating the AM code, the location and size of the module's dual port, and the bus interface type must be specified in the module's configuration for correct operation. See chapter 11 of the *Logicmaster 90-70 Programming Software User's Manual* (GFK-0263) for more information on configuration of third-party VME modules.

If the function is completed successfully, ok is set ON; otherwise, it is set OFF. It is also set OFF when:

- The number of data elements (N) is greater than the length (LEN) specified.
- The rack/slot value (R and S) is out of range or is not a valid VME location.
- The most significant byte of the dual port offset (OFF) is not zero.
- The most significant byte of the dual port address plus the dual port offset is not zero.
- Read beyond the end of dual port memory.
- Specified rack/slot not configured for a Third-Party VME module in **BUS INTERFACE** mode.
- If the dual port offset is an even number, configure for the odd byte only. If the dual port offset is an odd number, configure for word or single word.



Parameters:

Parameter	Description
enable	When the function is enabled, the data initialization is performed.
IN	IN contains the data to be written to the VME bus at the location defined by rack (R), slot (S), and dual port offset (OFF).
R	The rack number is specified in R.
S	The slot number is specified in S.
OFF	OFF specifies the dual port offset.
N	N contains the amount of data (data elements) to be written to the VME bus.
ok	The ok output is energized when the function is performed without error.
ST	The status word contains the status of the operation.
LEN	LEN is the length of the input array in bytes.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
IN	•	•	•	•	•	•	•		•	•	•	•	•	•		
R	•	•	•	•	•	•	•		•	•	•	•	•	•	•	
S	•	•	•	•	•	•	•		•	•	•	•	•	•	•	
OFF	•					•			•	•	•	•	•	•	•	
N	•	•	•	•	•	•	•		•	•	•	•	•	•	•	
ok	•															•
ST	•	•	•	•	•	†	•		•	•	•	•	•	•		

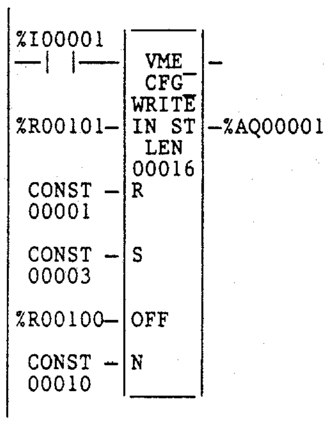
• Valid reference or place where power may flow through the function
† %SA, %SB, %SC only; %S cannot be used.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, when enable is ON, data from %R00101 through %R00110 of the array %R00101 through %R00116 is written to the VME bus at rack 1, slot 3 and dual port offset defined by %R00100. If an error was encountered, the status word %AQ0001 will contain an error code.



DATA_INIT (INT, UINT, DINT, WORD, DWORD, REAL)

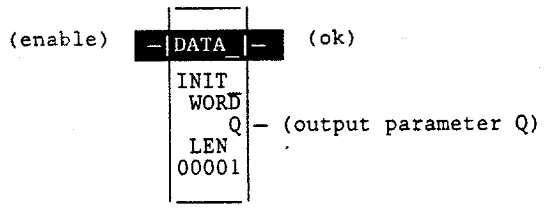
Use the Data Initialization (DATA_INIT) function to copy a block of constant data to a reference range.

The DATA_INIT function has one input parameter and two output parameters. When the function receives power flow, it copies the constant data to output Q. The function's constant data length (LEN) specifies how much constant data of the function type is copied to consecutive reference addresses starting at output Q.

Note

The output parameter is not included in coil checking.

The function passes power to the right whenever power is received.



Note

When the DATA_INIT instruction is first programmed, the constant data is initialized to zeroes. The constant data may be changed by zooming into the function (see below for details).

Parameters:

Parameter	Description
enable	When the function is enabled, the data initialization is performed.
ok	The ok output is energized whenever the function is enabled.
Q	When the data initialization is performed, the constant data is written to Q.
LEN	LEN specifies how much constant data is copied to consecutive reference addresses starting at output Q.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
ok	•															•
IN		o	o	o	o		o		•	•	•	•	•	•		

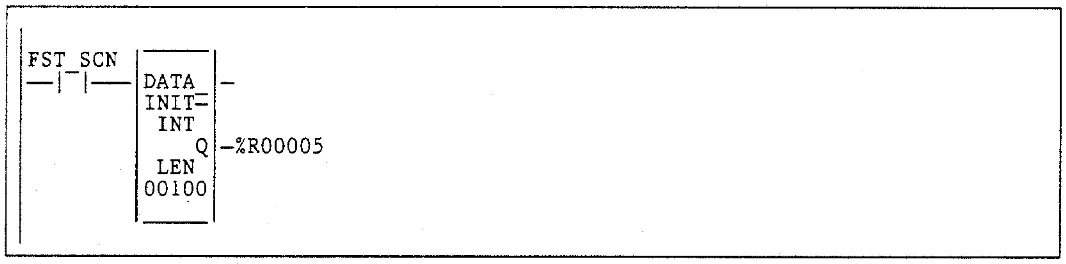
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for INT, UINT, or WORD data only..

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

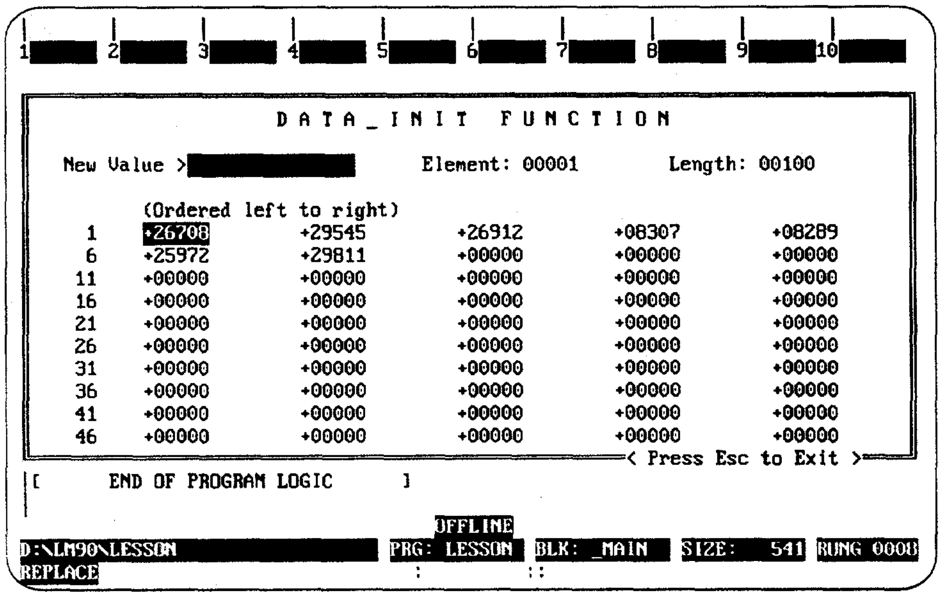
Example:

In the following example, on the first scan (FST_SCN) 100 words of initial data is copied to %R00005 through %R00104.



Zooming into the DATA_INIT_type Function Block

When you press **Zoom** (**F10**) to zoom into the DATA_INIT_type function block, a window similar to the one shown below is displayed.



```
 1     2     3     4     5     6     7     8     9     10

                      D A T A _ I N I T   F U N C T I O N

   New Value >                    Element: 00001        Length: 00100

         (Ordered left to right)
     1   +26708        +29545        +26912        +08307        +08289
     6   +25972        +29811        +00000        +00000        +00000
    11   +00000        +00000        +00000        +00000        +00000
    16   +00000        +00000        +00000        +00000        +00000
    21   +00000        +00000        +00000        +00000        +00000
    26   +00000        +00000        +00000        +00000        +00000
    31   +00000        +00000        +00000        +00000        +00000
    36   +00000        +00000        +00000        +00000        +00000
    41   +00000        +00000        +00000        +00000        +00000
    46   +00000        +00000        +00000        +00000        +00000
                                                   < Press Esc to Exit >
[       END OF PROGRAM LOGIC         ]

                                        OFFLINE
D:\LM90\LESSON                         PRG: LESSON  BLK: _MAIN  SIZE:    541 RUNG 0008
REPLACE                                     :           ::
```

The DATA_INIT_type window contains a *New Value* field which functions as a mini command line, an *Element* field which indicates which data element the cursor is currently positioned on, a *Length* field which tells how many elements the DATA_INIT_type instruction contains, and the block of constant data. The *Element* and *Length* fields cannot be edited.

To change a value, position the cursor on the element to be changed, and enter the new value in the *New Value* field.

To help you position the cursor, the window contains starting element line labels to the left of the data block. The *Element* field will contain the element number the cursor is currently positioned on.

The data block portion of the window will scroll and page.

DATA_INIT_COMM

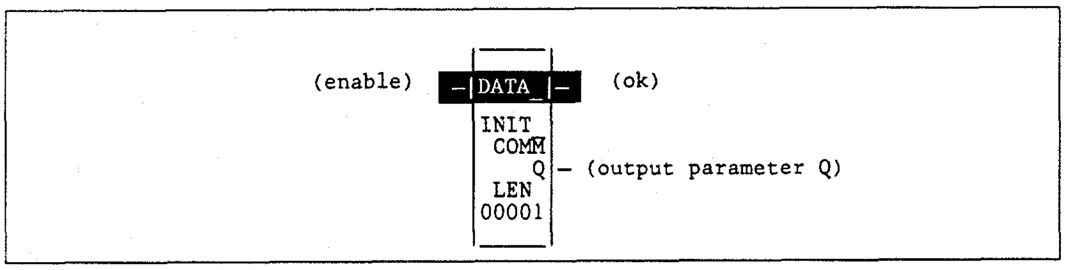
Use the Data Initialize Communications Request (DATA_INIT_COMM) function to initialize a COMMREQ function with a block of constant data. The IN parameter of the COMMREQ must correspond with output Q of this DATA_INIT_COMM function.

The DATA_INIT_COMM function has one input parameter and two output parameters. When the function receives power flow, it copies the constant data to output Q. The function's constant data length (LEN) specifies how many words of constant data are to be copied to consecutive reference addresses starting at output Q. The length should be equal to the size of the COMMREQ function's entire command block.

Note

The output parameter is not included in coil checking.

The function passes power to the right whenever power is received.



Note

When the DATA_INIT instruction is first programmed, the constant data is initialized to zeroes. The constant data may be changed by zooming into the function (see below for details).

Parameters:

Parameter	Description
enable	When the function is enabled, the data initialization is performed.
ok	The ok output is energized whenever the function is enabled.
Q	When the data initialization is performed, the constant data is written to Q.
LEN	LEN specifies how many words of constant data are to be copied to consecutive reference addresses starting at output Q. LEN must equal the size of the COMMREQ function's entire command block.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
ok	•															•
Q									•	•	•	•	•	•		

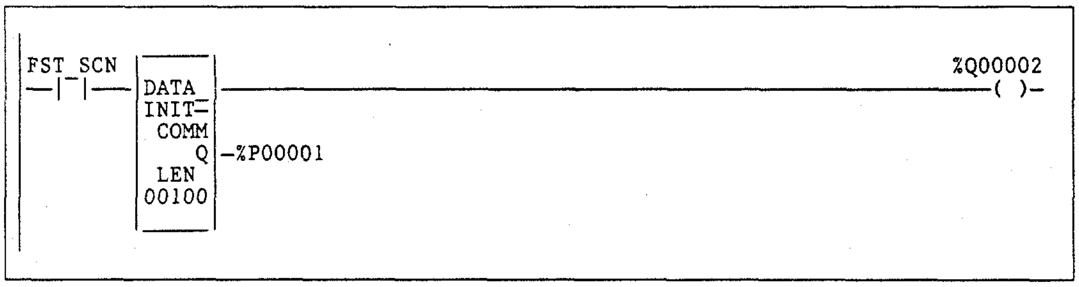
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

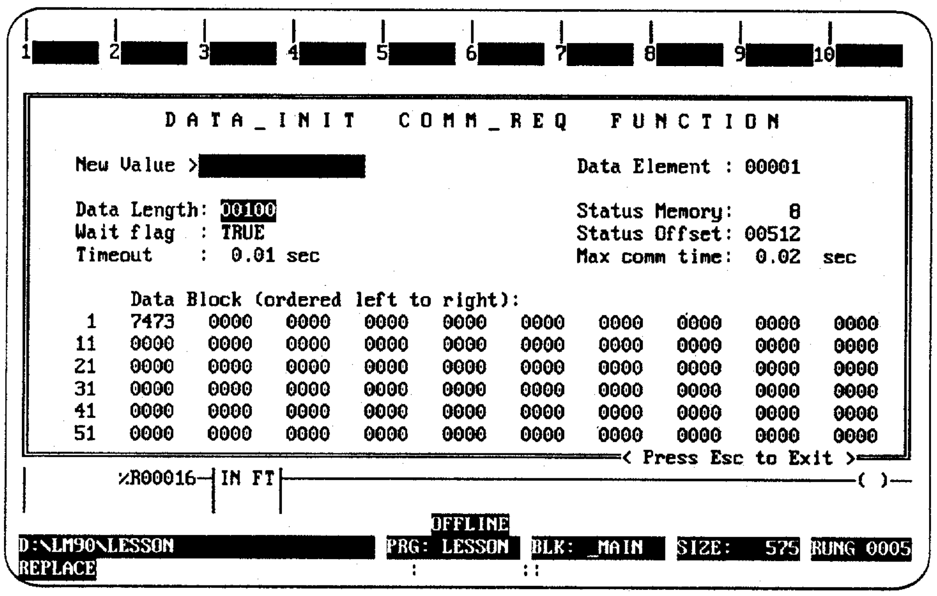
Example:

In the following example, on the first scan (FST_SCN) a command block consisting of 100 words of data and 6 words of header is copied to %P00001 through %P00099; and %Q00002 will receive power.



Zooming into the DATA_INIT_COMM Function Block

When you press **Zoom (F10)** to zoom into the DATA_INIT_COMM function block, a window similar to the one shown below is displayed.



```
 1      2      3      4      5      6      7      8      9      10

            D A T A _ I N I T   C O M M _ R E Q   F U N C T I O N

     New Value >                                 Data Element : 00001

     Data Length: 00100                          Status Memory:     8
     Wait flag  : TRUE                           Status Offset: 00512
     Timeout    :  0.01 sec                      Max comm time:  0.02  sec

            Data Block (ordered left to right):
       1    7473  0000  0000  0000  0000  0000  0000  0000  0000  0000
      11    0000  0000  0000  0000  0000  0000  0000  0000  0000  0000
      21    0000  0000  0000  0000  0000  0000  0000  0000  0000  0000
      31    0000  0000  0000  0000  0000  0000  0000  0000  0000  0000
      41    0000  0000  0000  0000  0000  0000  0000  0000  0000  0000
      51    0000  0000  0000  0000  0000  0000  0000  0000  0000  0000
                                                 < Press Esc to Exit >
        %R00016-|IN FT|-------------------------------------------------( )-

                                          OFFLINE
D:\LM90\LESSON                        PRG: LESSON  BLK: _MAIN  SIZE:   575  RUNG 0005
REPLACE
```

The DATA_INIT_COMM window contains a *New Value* field for entering values and a *Data Element* field, which indicates which data element in the data block the cursor is currently positioned on. Only the *Data Element* field cannot be changed.

The window also contains the first six words of the COMMREQ command block, with labels consistent with each word's use and the COMMREQ data block. The first six words are *Data Length, Wait flag, Timeout, Status Memory, Status Offset, and Max comm time*.

The line labels to the left of the data block elements and the *Data Element* field will correspond to the data block portion of the COMMREQ block. For example, the data block elements will be labeled beginning with number 1 even though the data block begins at the seventh word of the instruction.

The data block portion of the window will scroll and page.

Note

The data block information is in hexadecimal.

DATA_INIT_ASCII

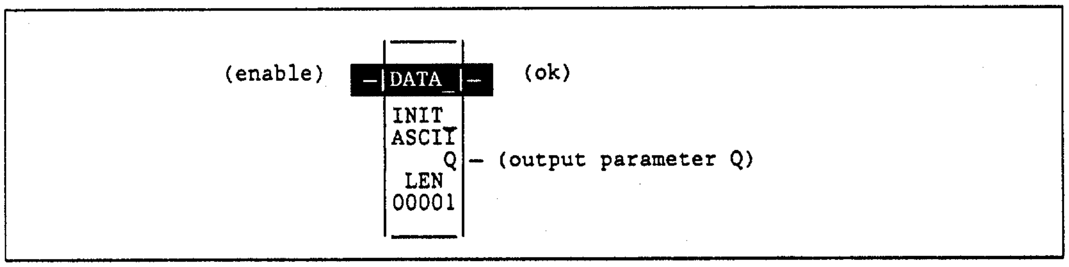
Use the Data Initialize ASCII (DATA_INIT_ASCII) function to copy a block of constant ASCII text to a reference range.

The DATA_INIT_ASCII function has one input parameter and two output parameters. When the function receives power flow, it copies the constant data to output Q. The function's constant data length (LEN) specifies how many bytes of constant text are copied to consecutive reference addresses starting at output Q. LEN must be an even number.

Note

The output parameter is not included in coil checking.

The function passes power to the right whenever power is received.



Note

When the DATA_INIT instruction is first programmed, the constant data is initialized to blanks. The constant data may be changed by zooming into the function (see below for details).

Parameters:

Parameter	Description
enable	When the function is enabled, the data initialization is performed.
ok	The ok output is energized whenever the function is enabled.
Q	When the data initialization is performed, the constant data is written to Q.
LEN	LEN specifies how many bytes of constant text are copied to consecutive reference addresses starting at output Q. LEN must be an even number.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
ok	•															•
Q		•	•	•	•		•		•	•	•	•	•	•		

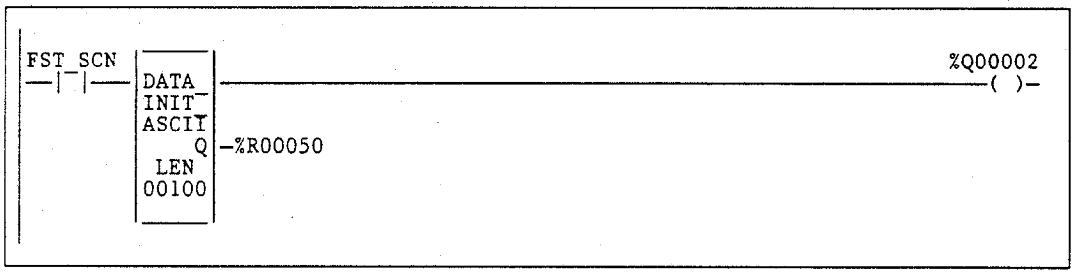
Note Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, on the first scan (FST_SCN) the decimal equivalent of 100 bytes of ASCII text is copied to %R00050 through %R00149; and %Q00002 will receive power.

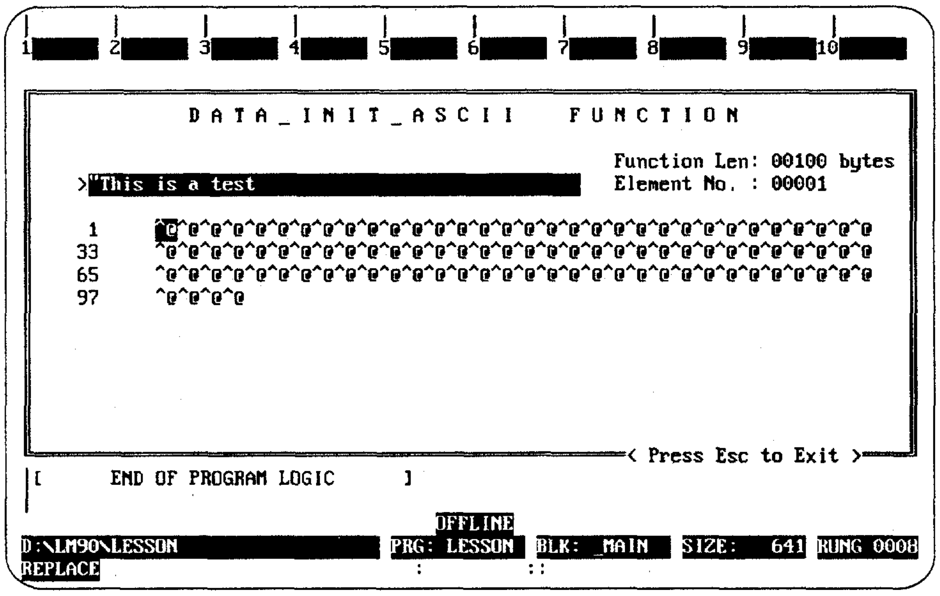


Zooming into the DATA_INIT_ASCII Function Block

When you press **Zoom (`F10`)** to zoom into the DATA_INIT_ASCII function block, a window similar to those shown below is displayed.

The DATA_INIT_ASCII window contains a mini command line, a ***Function Len*** field which indicates how many character bytes the instruction contains, an ***Element No.*** field which indicates which character byte the cursor is currently positioned on, and the block of ASCII text.

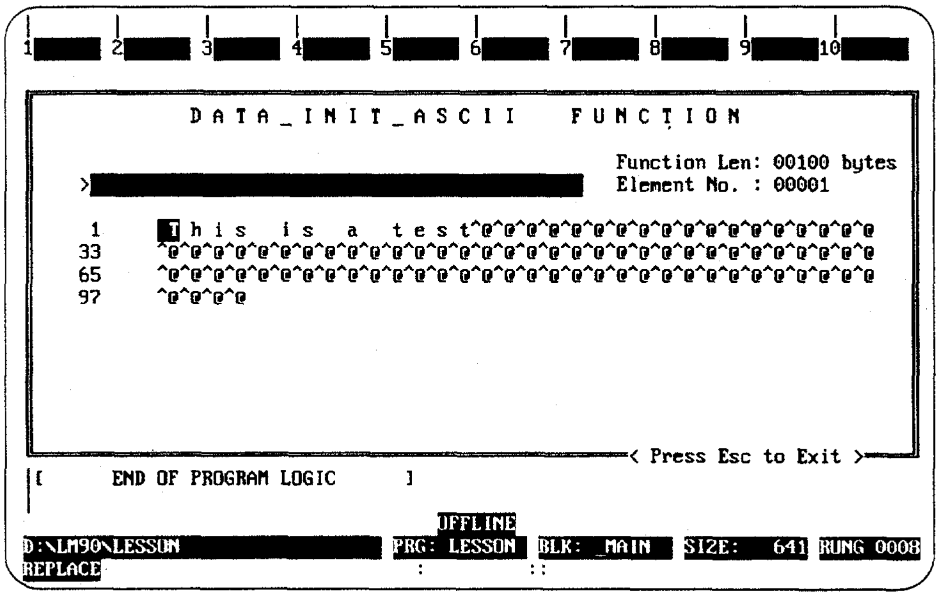
ASCII text is entered by positioning the cursor on the starting byte and typing a quotation mark (") followed by the desired string on the mini command line (e.g., "This is a test). (Refer to the example on the next page.)



```
 1      2      3      4      5      6      7      8      9      10

            D A T A _ I N I T _ A S C I I    F U N C T I O N

                                               Function Len: 00100 bytes
     >"This is a test                          Element No. : 00001

      1     ^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
     33     ^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
     65     ^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
     97     ^@^@^@^@

                                               < Press Esc to Exit >
|[      END OF PROGRAM LOGIC      ]
                                        OFFLINE
D:\LM90\LESSON                      PRG: LESSON  BLK: _MAIN  SIZE:   641 RUNG 0008
REPLACE                                 :         ::
```

You can enter as many characters as will fit on the command line at one time. Enter non-printable characters with a backslash immediately followed by the 3-digit decimal equivalent of the character (e.g., "\010 for line feed).

After entering the text on the command line, press the **Enter** key.



```
 1      2      3      4      5      6      7      8      9      10

            D A T A _ I N I T _ A S C I I    F U N C T I O N

                                               Function Len: 00100 bytes
     >                                         Element No. : 00001

      1     T h i s   i s   a   t e s t^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
     33     ^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
     65     ^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
     97     ^@^@^@^@

                                               < Press Esc to Exit >
|[      END OF PROGRAM LOGIC      ]
                                        OFFLINE
D:\LM90\LESSON                      PRG: LESSON  BLK: _MAIN  SIZE:   641 RUNG 0008
REPLACE                                 :         ::
```

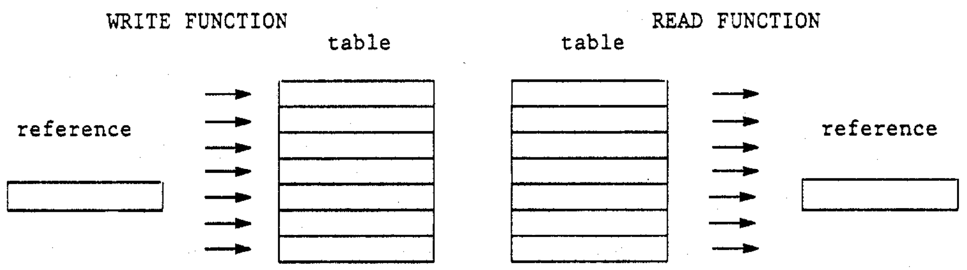
Press the **Escape** key to exit this window and return to the ladder diagram logic display.

Section 7: Data Table Functions

Data tables provide automatic data move capabilities. The data table functions are used to enter values into or copy values out of a table. To use these functions properly, the program must include other logic to move data through the tables and to initialize table pointers.

Moving Values In and Out of a Table

All of the table write functions fill a table with values that are read from a specified reference location. Similarly, all of the table read functions copy values from a table to a specified reference location.



One value is written or read each time these functions are called. Other logic in the program must be used to place new values in the reference for a write function, or to capture values from the reference after a read function.

Initializing the Table Pointer

The read and write functions keep track of the current table location by means of a pointer. Additional program logic must be used to initialize this value so that the function will begin reading or writing the table at the correct location. Normally, the value in this reference is initialized to zero.

This section describes the following data table functions.

Abbreviation	Function	Description	Page
TBLRD	Table Read	Copy a value from a specified table location to an output reference.	4-126
TBLWR	Table Write	Copy a value from an input reference to a specified table location.	4-128
LIFORD	LIFO Read	Remove the entry at the pointer location, and decrement the pointer by one. LIFORD is used in conjunction with LIFOWRT (see below).	4-130
LIFOWRT	LIFO Write	Increment the table pointer and write data to the table. LIFOWRT is used in conjunction with LIFORD (see above).	4-132
FIFORD	FIFO Read	Remove the entry at the bottom of the table, and decrement the pointer by one. FIFORD is used in conjunction with FIFOWRT (see below).	4-134
FIFOWRT	FIFO Write	Increment the table pointer and write data to the table. FIFOWRT is used in conjunction with FIFORD (see above).	4-136
SORT	Sort	Sort an array in ascending order.	4-138
ARRAY_MOVE	Array Move	Copy a specified number of data elements from a source array to a destination array.	4-140
SRCH_EQ	Search Equal	Search for all array values equal to a specified value.	4-144
SRCH_NE	Search Not Equal	Search for all array values not equal to a specified value.	4-144
SRCH_GT	Search Greater Than	Search for all array values greater than a specified value.	4-144
SRCH_GE	Search Greater Than or Equal	Search for all array values greater than or equal to a specified value.	4-144
SRCH_LT	Search Less Than	Search for all array values less than a specified value.	4-144
SRCH_LE	Search Less Than or Equal	Search for all array values less than or equal to a specified value.	4-144
ARRAY_RANGE	Array Range	Determine if a value is between the range specified in two tables.	4-147

All values in the table must be the same type, which may be signed integer (INT), double precision signed integer (DINT), unsigned integer (UINT), word (WORD), or double word (DWORD). The BYTE data type is available for SRCH_EQ, SRCH_NE, SRCH_GT, SRCH_LT, SRCH_GE, SRCH_LE, and ARRAY_MOVE; and BIT is also available for ARRAY_MOVE.

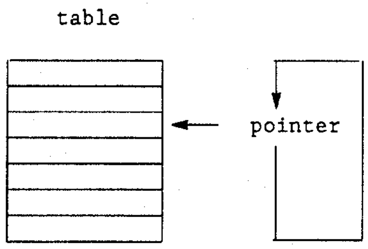
Note

No REAL data type exists for the data table functions; however, the DWORD data type may be used to manipulate real data.

The maximum length allowed for each data table function is 32,767 elements, except for the SORT function which is restricted to 64 elements.

TBLRD (INT, UINT, DINT, WORD, DWORD)

The Table Read (TBLRD) function is used to sequentially read values in a table which never becomes full. When the pointer reaches the end of the table, it automatically wraps around to the beginning of the table.



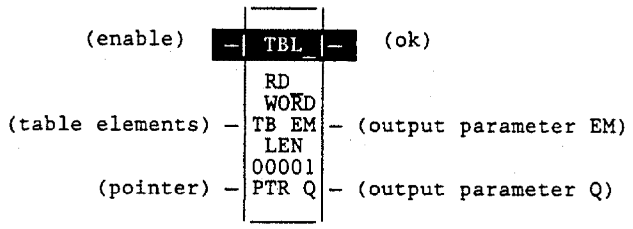
1. The function increments the pointer by one.
2. The function copies data indicated by the pointer to output parameter Q. Additional program logic must then be used to capture the data from the output reference.
3. Steps 1 and 2 are repeated each time the instruction is executed, until the table is empty (PTR = LEN).
4. When the table is full, the pointer wraps around to the beginning of the table.

Note

TBLRD and TBLWRT functions can operate on the same or different tables. By specifying a different reference for the pointer, these functions can access the same data table at different locations or at different rates.

The TBLRD function has three input parameters and three output parameters. When the function receives power flow, the pointer (PTR) increments by one. If this new pointer location is the last item in the table, the output EM is set ON. The next time the function executes, the pointer will automatically be set back to 1. After the pointer is incremented, the content at the new pointer location is copied to output Q.

The function always passes power to the right when power is received.



Parameters:

Parameter	Description
enable	When enable is energized, the table read is performed.
TB	TB contains the elements in the table.
PTR	PTR is incremented; it then points to the next table element to be read.
ok	The ok output is energized whenever the function is enabled.
EM	Output EM is energized when the last element of the table is read.
Q	Output Q contains the element read from the table.
LEN	LEN must be between 1 and 32,767.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
TB	•	o	o	o	o	Δ†	o		•	•	•	•	•	•		
PTR		•	•	•	•		•		•	•	•	•	•	•		
ok	•															•
EM	•															•
Q	•	o	o	o	o	Δo	o		•	•	•	•	•	•		

Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for INT, UINT, or WORD data only; not valid for DINT or DWORD.
Δ Valid reference for WORD data only; not valid for INT, UINT, DINT, or DWORD.
† %SA, %SB, %SC only; %S cannot be used.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

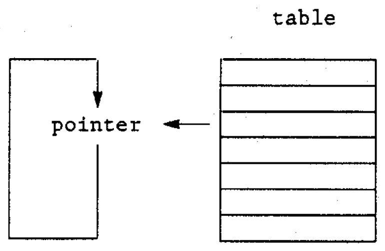
Example:

In the following example, WIDGETS is a table with 20 integer elements. When the enabling input %M00346 is ON, the pointer increments and the contents of the next element of the table is copied into ITEM_CT. %L00001 functions as the pointer into the data table. %M01001 is used to signal when all items of the data table have been accessed.

```
|%M00346 |_____|
|--| |---| TBL_|-
|        | RD  |
|        | INT |                                              %M01001
|WIDGETS-|TB EM|----------------------------------------------( )-
|        | LEN |
|        |00020|
|%L00001-|PTR Q|-ITEM_CT
|        |_____|
|
```

TBLWRT (INT, UINT, DINT, WORD, DWORD)

The Table Write (TBLWRT) function is used to sequentially update values in a table that never becomes full. When the pointer reaches the end of the table, it automatically returns to the beginning of the table.



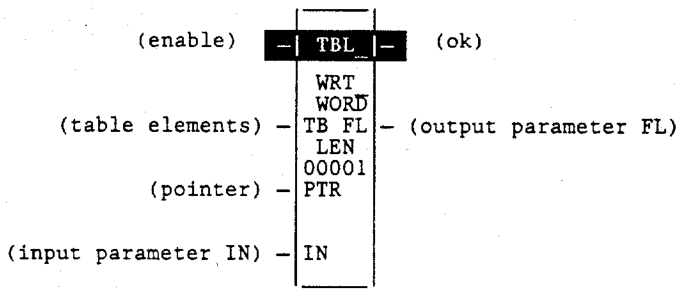
1. The function increments the pointer by one.
2. The function copies data from input parameter IN to the position in the table indicated by the pointer. (It will write over any value currently at that location.) Additional program logic must then be used to place the data in the input reference.
3. Steps 1 and 2 are repeated each time the instruction is executed, until the table is full (PTR = LEN).
4. When the table is full, the pointer wraps around to the beginning of the table.

Note

TBLWRT and TBLRD functions can operate on the same or different tables. By specifying a different reference for the pointer, these functions can access the same data table at different locations or at different rates.

The TBLWRT function has four input parameters and two output parameters. When the function receives power flow, the pointer (PTR) increments by 1. If this new pointer location is the last item in the table, the output FL is set to ON. The next time the function executes, the pointer will automatically be set back to 1. After incrementing the pointer, the TBLWRT function writes the content of the input reference to the current pointer location, overwriting data already stored there.

The function always passes power to the right when power is received.



Parameters:

Parameter	Description
enable	When enable is energized, the table write is performed.
TB	TB contains the elements in the table.
PTR	PTR is incremented; it then points to the next table element to be written.
IN	IN contains the element to be written to the table.
ok	The ok output is energized whenever the function is enabled.
FL	Output FL is energized when IN is written to the last element of the table.
LEN	LEN must be between 1 and 32,767.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
TB		o	o	o	o	Δ†	o		•	•	•	•	•	•		
PTR		•	•	•	•		•		•	•	•	•	•	•		
IN	•	o	o	o	o	Δ†	o		•	•	•	•	•	•	•	
ok	•															•
FL	•															•

Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for INT, UINT, or WORD data only; not valid for DINT or DWORD.
Δ Valid reference for WORD data only; not valid for INT, UINT, DINT, or DWORD.
† %SA, %SB, %SC only; %S cannot be used.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

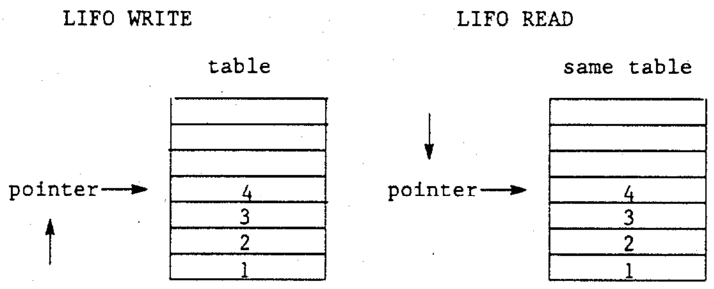
Example:

In the following example, WIDGETS is a table with 20 integer elements. When the enabling input %I00012 is ON, the pointer increments and the contents of %P00077 are written into the table at the pointer location. %L00001 functions as the pointer into the data table.

```
|%I00012  _______
|--| |---| TBL_  |-
|         | WRT  |
|         | INT  |                                          %M01001
|WIDGETS-|TB  FL|--------------------------------------------( )-
|         | LEN  |
|         |00020 |
|%L00001-|PTR   |
|         |      |
|%P00077-|IN    |
|         |______|
```

LIFORD (INT, UINT, DINT, WORD, DWORD)

The Last-In-First-Out (LIFO) Read (LIFORD) function is used to move data out of tables. Values are always moved out of the top of the table. If the pointer reaches the last location and the table becomes full, the LIFORD function must be used to remove the entry at the pointer location and decrement the pointer by one. The LIFORD function is used in conjunction with the LIFOWRT function, which increments the pointer and writes entries into the table.

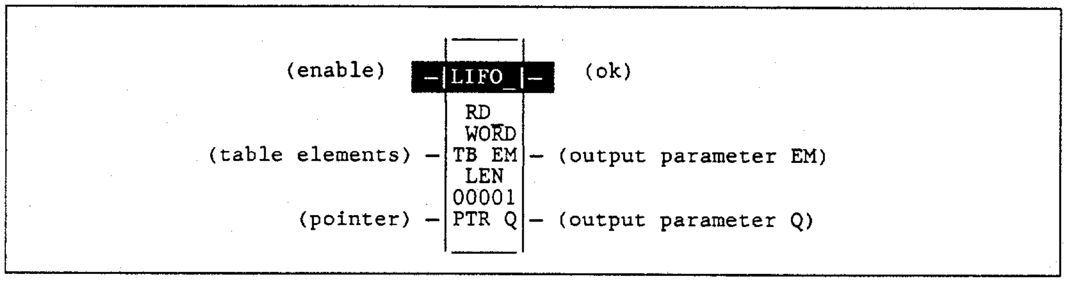


Sequence of events:

1. The function copies data indicated by the pointer to output parameter Q. Additional program logic must then be used to place the data in the input reference.
2. The function decrements the pointer by one.
3. Steps 1 and 2 are repeated each time the instruction is executed, until the table is empty (PTR = LEN).
4. The pointer does not wrap around when the table is full.

The LIFORD function has three input parameters and three output parameters. When the function receives power flow, the data at the pointer location is copied to output Q, then the pointer is decremented. If this causes the pointer location to become 0, the output EM is set ON. Therefore, EM indicates whether or not the table is empty. If the table is empty when the function receives power flow, no read will occur. The pointer always indicates the last item entered into the table.

The function passes power to the right if the pointer was in range for an element to be read.



Parameters:

Parameter	Description
enable	When enable is energized, the read operation is performed.
TB	TB contains the elements in the table.
PTR	PTR points to the next LIFO element to be read; after the read, it is decremented.
ok	The ok output is energized when EN is ON and 0 < PTR ≤ LEN.
EM	Output EM is energized when the last element is read.
Q	Output Q contains the element read from the table.
LEN	LEN must be between 1 and 32,767.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
TB	•	o	o	o	o	o	Δ†	o		•	•	•	•	•	•	
PTR		•	•	•	•		•		•	•	•	•	•	•		
ok	•															•
EM	•															•
Q	•	o	o	o	o	Δ†	o		•	•	•	•	•	•		•

Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for INT, UINT, or WORD data only; not valid for DINT or DWORD.
Δ Valid reference for WORD data only; not valid for INT, UINT, DINT, or DWORD.
† %SA, %SB, %SC only; %S cannot be used.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

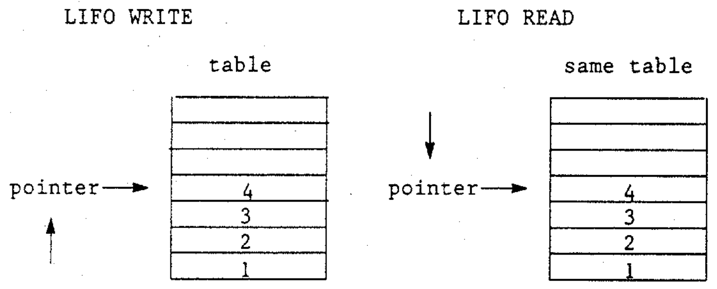
Example:

In the following example, PRODUCT is a LIFO table with 100 word-sized elements. When the enabling input PACK_IT is ON, the data item at the top of the table is copied into the reference indicated by the nickname CART. The reference identified by STK_PTR contains the table pointer. Output coil EMPTY indicates when the table is empty.

```
|PACK_IT  _______
|--| |---|LIFO_  |-
|          RD
|          WORD                                                   EMPTY
|PRODUCT-|TB  EM|--------------------------------------------------( )-
|          LEN
|         00100
|STK_PTR-|PTR  Q|-CART
|         _______
|
```

LIFOWRT (INT, UINT, DINT, WORD, DWORD)

The Last-In-First-Out (LIFO) Write (LIFOWRT) function is used to increment the table pointer by one, and then add an entry above the pointer location in a table. Values are always moved in at the top of the table. If the pointer reaches the last location and the table becomes full, no further values can be added by the LIFOWRT function. The LIFORD function must then be used to remove the entry at the pointer location and decrement the pointer by one.

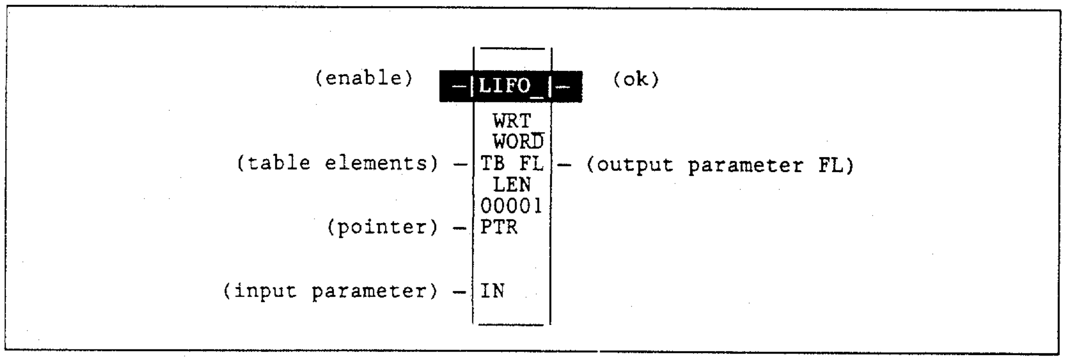


Sequence of events:

1. The function increments the table pointer by one.
2. The function copies data from input parameter IN to the position in the table indicated by the pointer. (It will write over any value currently at that location.) Additional program logic must then be used to place the data in the input reference.
3. Steps 1 and 2 are repeated each time the instruction is executed, until the table is full (PTR = LEN).
4. The pointer does not wrap around when the table is full.

The LIFOWRT function has four input parameters and two output parameters. When the function receives power flow, the pointer increments by 1; then the new data is written at the pointer location. If the pointer was already at the last location in the table, no data is written and the function does not pass power to the right. The pointer always indicates the last item entered into the table. If the table is full, it is not be possible to add more entries to it.

The function passes power to the right after a successful execution.



Parameters:

Parameter	Description
enable	When enable is energized, the write operation is performed.
TB	TB contains the elements in the table.
PTR	PTR is incremented; it then points to the next element to be written.
IN	IN contains the element to be written to the table.
ok	The ok output is energized when the function is enabled and PTR < LEN.
FL	Output FL is energized when the last element has been written.
LEN	LEN must be between 1 and 32,767.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
TB		o	o	o	o	Δ÷	o		•	•	•	•	•	•		
PTR		•	•	•	•		•		•	•	•	•	•	•		
IN	•	o	o	o	o	Δ÷	o		•	•	•	•	•	•	•	
ok	•															•
FL	•															•

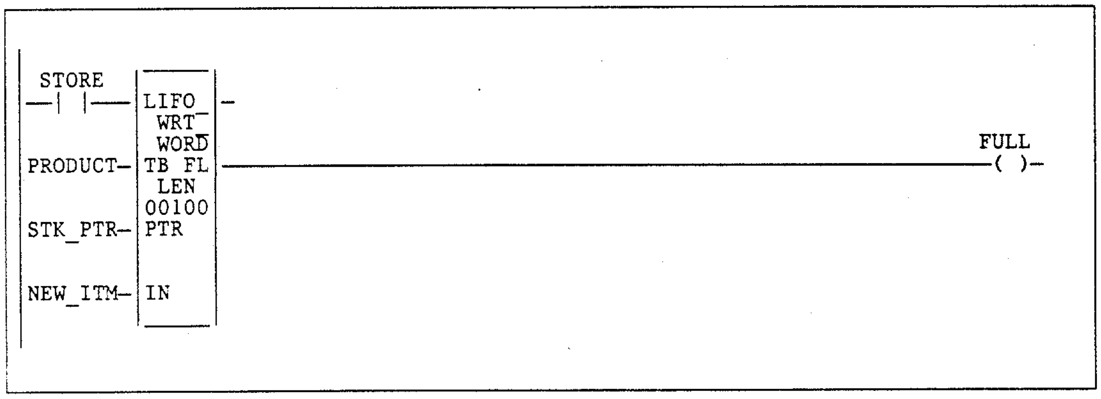
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for INT, UINT, or WORD data only; not valid for DINT or DWORD.
Δ Valid reference for WORD data only; not valid for INT, UINT, DINT, or DWORD.
÷ %SA, %SB, %SC only; %S cannot be used.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

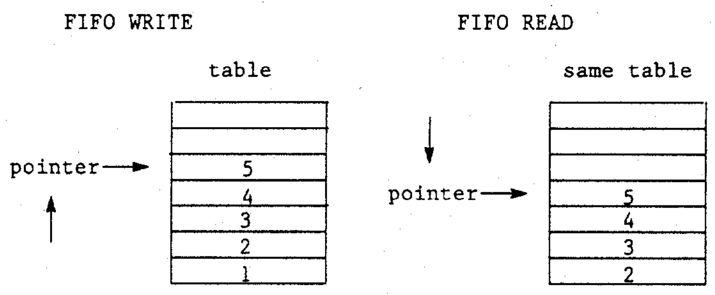
Example:

In the following example, PRODUCT is a LIFO table with 100 word-sized elements. When the enabling input STORE is ON, a data item from NEW_ITEM is copied to the table location pointed to by the value in STK_PTR. Output node FL will pass power when PTR = LEN, firing the FULL coil. No further data from NEW_ITEM can be added to the table without first copying data out, using the LIFORD function.



FIFORD (INT, UINT, DINT, WORD, DWORD)

The First-In-First-Out (FIFO) Read (FIFORD) function is used to move data out of tables. Values are always moved out of the bottom of the table. If the pointer reaches the last location and the table becomes full, the FIFORD function must be used to remove the entry at the pointer location and decrement the pointer by one. The FIFORD function is used in conjunction with the FIFOWRT function, which increments the pointer and writes entries into the table.

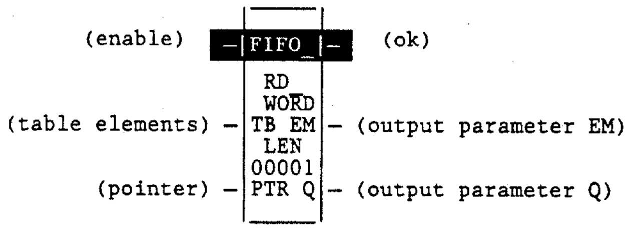


Sequence of events:

1. The function copies the top location of the table to output parameter Q. Additional program logic must then be used to place the data in the input reference.
2. The remaining items in the table are copied to a lower numbered position in the table.
3. The function decrements the pointer by one.
4. Steps 1 and 2 are repeated each time the instruction is executed, until the table is empty (PTR = LEN).
5. The pointer does not wrap around when the table is full.

The FIFORD function has three input parameters and three output parameters. When the function receives power flow, the data at the first location of the table is copied to output Q. Next, each item in the table is moved down to the next lower location. This begins with item 2 in the table, which is moved into position 1. Finally, the pointer is decremented. If this causes the pointer location to become 0, the output EM is set ON. Therefore, EM indicates whether or not the table is empty.

The FIFORD function passes power to the right if the pointer is greater than zero and less than the value specified for LEN.



Parameters:

Parameter	Description
enable	When enable is energized, the FIFO read is performed. It is always the first element of the FIFO table that is read.
TB	TB contains the elements of the FIFO table.
PTR	PTR points to the last element of the FIFO table.
ok	The ok output is energized when the function is enabled and 0 < PTR < LEN.
EM	Output EM is energized when the final element is read.
Q	Output Q contains the element read from the FIFO table.
LEN	LEN must be between 1 and 32,767.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
TB	•	o	o	o	o	Δ†	o		•	•	•	•	•	•		
PTR		•	•	•	•		•		•	•	•	•	•	•		
ok	•															•
EM	•															•
Q	•	o	o	o	o	Δ†	o		•	•	•	•	•	•		

Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for INT, UINT, or WORD data only; not valid for DINT or DWORD.
Δ Valid reference for WORD data only; not valid for INT, UINT, DINT, or DWORD.
† %SA, %SB, %SC only; %S cannot be used.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

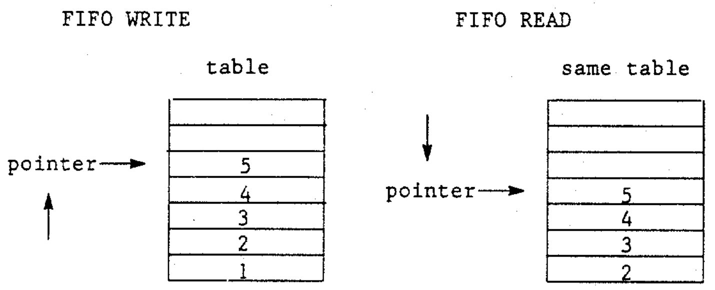
Example:

In the following example, PRODUCT is a FIFO table with 100 word-sized elements. When the enabling input PACK_IT is ON, the PRODUCT data item in the table location pointed to by STK_PTR is copied to the reference location specified in CART. This table location pointed to would be the bottom, or oldest data item in the table. The number in STK_PTR will be decremented. A copy of the oldest data item in the PRODUCT table will be left behind in each table location as the current data is copied out during successive PACK_IT triggers. Output node EM will pass power when the PTR = 0, firing the coil EMPTY. No further data from the PRODUCT table can be read without first copying data in using the FIFOWRT function.

```
|PACK_IT  |_____|
|--| |----|FIFO_|-
|         | RD  |
|         | WORD|                                                    EMPTY
|PRODUCT--|TB EM|----------------------------------------------------( )-
|         | LEN |
|         |00100|
|STK_PTR--|PTR Q|-CART
|         |_____|
```

FIFOWRT (INT, UINT, DINT, WORD, DWORD)

The First-In-First-Out (FIFO) Write (FIFOWRT) function is used to increment the table pointer by one, and then add an entry at the new pointer location in a table. Values are always moved in at the bottom of the table. If the pointer reaches the last location and the table becomes full, no further values can be added by the FIFOWRT function. The FIFORD function must then be used to remove the entry at the pointer location and decrement the pointer by one.

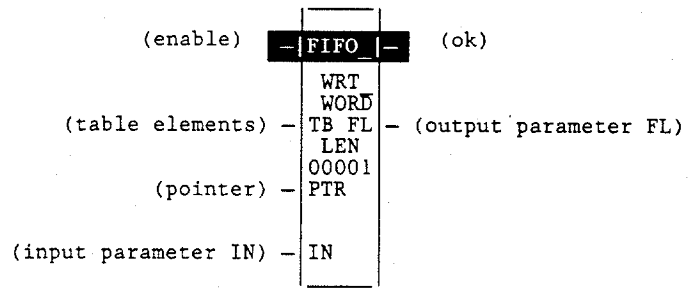


Sequence of events:

1. The function increments the pointer by one.
2. The function copies data from input parameter IN to the position in the table indicated by the pointer. (It will write over any value currently at that location.) Additional program logic must then be used to place the data in the input reference.
3. Steps 1 and 2 are repeated each time the instruction is executed, until the table is full (PTR = LEN).
4. The pointer does not wrap around when the table is full.

The FIFOWRT function has four input parameters and two output parameters. When the function receives power flow, the pointer is incremented by 1. Then, input data is written into the table at the pointer location. If the pointer was already at the last location in the table, no data is written and the function does not pass power to the right. The pointer always indicates the last item entered into the table. If the table becomes full, it will not be possible to add more entries to it.

The FIFOWRT function passes power to the right after a successful execution.



Parameters:

Parameter	Description
enable	When enable is energized, the write operation is performed.
TB	TB contains the elements of the FIFO table.
PTR	PTR points to the last element of the FIFO table.
IN	IN contains the value to be written to the FIFO table.
ok	The ok output is energized when the function is enabled and PTR < LEN.
FL	Output FL is energized when the last element position of the FIFO table is written to.
LEN	LEN must be between 1 and 32,767.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
TB		o	o	o	o	Δ†	o		•	•	•	•	•	•		
PTR		•	•	•	•		•		•	•	•	•	•	•		
IN	•	o	o	o	o	Δ†	o		•	•	•	•	•	•	•	
ok	•															•
FL	•															•

Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).

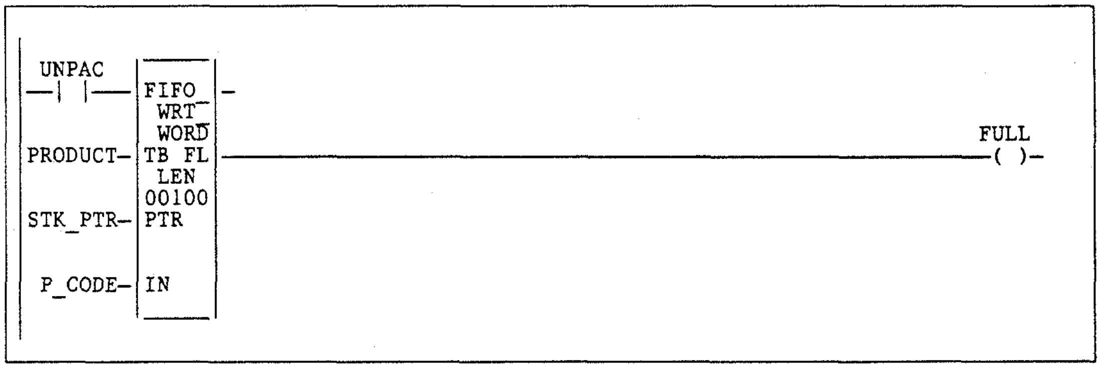
•	Valid reference or place where power may flow through the function.
o	Valid reference for INT, UINT, or WORD data only; not valid for DINT or DWORD.
Δ	Valid reference for WORD data only; not valid for INT, UINT, DINT, or DWORD.
†	%SA, %SB, %SC only; %S cannot be used.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, PRODUCT is a FIFO table with 100 word-sized elements. When the enabling input UNPACK is ON, a data item from P_CODE is copied to the table location pointed to by the value in STK_PTR. Output node FL will pass power when PTR = LEN, firing the FULL coil. No further data from P_CODE can be added to the table without first copying data out, using the FIFORD function.



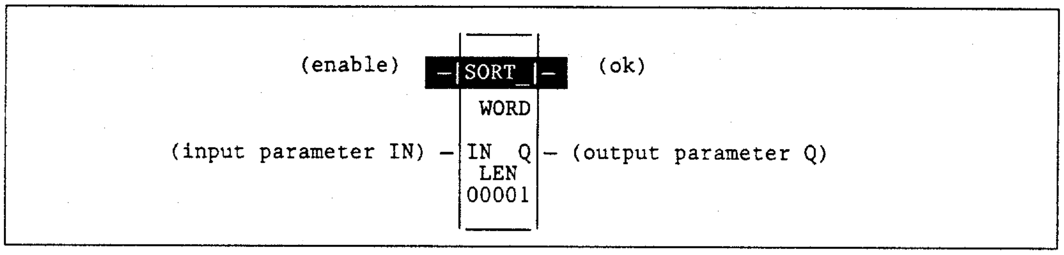
SORT (INT, UINT, WORD)

The SORT function is used to sort an array in ascending order. The function has two input and two output parameters. EN is a boolean enable; IN is the array to be sorted. OK is a boolean output providing power flow, and Q is an array of integers that gives the index that the sorted elements had in the original array or list. Q is exactly the same size as IN. It also has a specification (LEN) of the number of elements to be sorted.

The SORT function is restricted to operate on arrays with up to 64 elements. When EN is ON, all of the elements of IN are sorted into ascending order, based on their type. The array Q is also created, giving the original position that each sorted element held in the unsorted array. OK is always set ON.

Note

Do not use the SORT function in a timed or triggered input program block.



Parameters:

Parameter	Description
enable	When enable is energized, the sort is performed.
IN	IN contains the array of elements to be sorted. At the conclusion of the sort, the elements of IN are in sorted order.
ok	The ok output is energized whenever LEN is less than 65. However, the programmer does not limit the LEN parameter to 64. If LEN is greater than 64, the function block operates only on the first 64 units of LEN.
Q	Output Q contains an array of indexes that gives the position of the sorted elements in the original array.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
IN		•	•	•	•		•		•	•	•	•	•	•		
ok	•															•
Q	•	•	•	•	•		•		•	•	•	•	•	•		

Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.

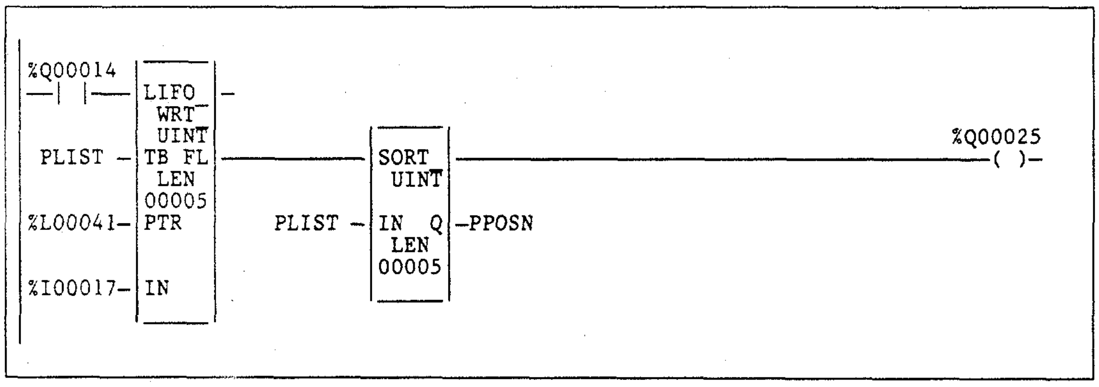
Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, new part numbers (%I00017 - %I00032) are pushed onto a parts array PLIST every time %Q00014 is ON. When the array is filled, it is sorted and the output %Q00025 is turned on. The array PPOSN will contain the original position that the now-sorted elements held before the sort was done on PLIST.

If PLIST was an array of five elements and contained the values 25, 67, 12, 35, 14 before the sort, then after the sort it would contain the values 12, 14, 25, 35, 67. PPOSN would contain the values 3, 5, 1, 4, 2.



ARRAY_MOVE (INT, UINT, DINT, BIT, BYTE, WORD, DWORD)

Use the Array Move (ARRAY_MOVE) function to copy a specified number of data elements from a source array to a destination array.

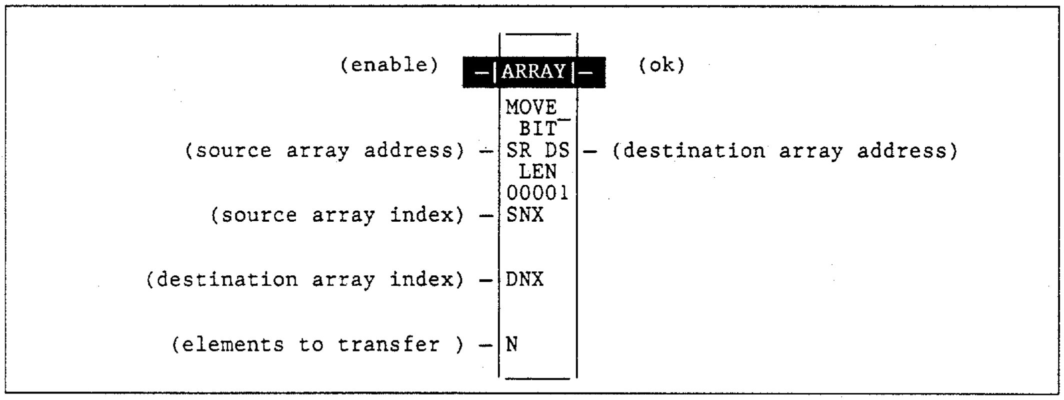
The ARRAY_MOVE function has six input parameters and two output parameters. When the function receives power flow, the number of data elements in the count indicator (N) is extracted from the input array starting with the indexed location (SR + SNX − 1). The data elements are written to the output array starting with the indexed location (DS + DNX − 1). The LEN operand specifies the number of elements that make up each array.

For ARRAY_MOVE_BIT, when word-oriented memory is selected for the parameters of the source array and/or destination array starting address, the least significant bit of the specified word is the first bit of the array. The value displayed contains 16 bits, regardless of the length of the array.

The indices in an ARRAY_MOVE instruction are 1-based. In using an ARRAY_MOVE, no element outside either the source or destination arrays, as specified by their starting address and length, may be referenced.

The ok output will receive power flow, unless one of the following conditions occurs:

- Enable is OFF.
- (N + SNX) is greater than LEN.
- (N + DNX) is greater than LEN.



Parameters:

Parameter	Description
enable	When the function is enabled, the operation is performed.
SR	SR contains the starting address of the source array. For ARRAY_MOVE_BIT, any reference may be used; it does not need to be byte aligned. However, 1 bit, beginning with the reference address specified, is displayed online.
SNX	SNX contains the index of the source array.
DNX	DNX contains the index of the destination array.
N	N provides a count indicator.
ok	The ok output is energized whenever the function is enabled. If NX is out of range, ok will not be energized.
DS	DS contains the starting address of the destination array. For ARRAY_MOVE_BIT, any reference may be used; it does not need to be byte aligned. However, 1 bit, beginning with the reference address specified, is displayed online.
LEN	LEN specifies the number of elements starting at SR and DS that make up each array.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
SR	•	o	o	o	o	Δ†	o	o	•	•	•	•	•	•		
SNX	•	•	•	•	•		•		•	•	•	•	•	•	•	
DNX	•	•	•	•	•		•		•	•	•	•	•	•	•	
N	•	•	•	•	•		•		•	•	•	•	•	•	•	
DS	•	o	o	o	o	†	o		•	•	•	•	•	•		
ok	•															•

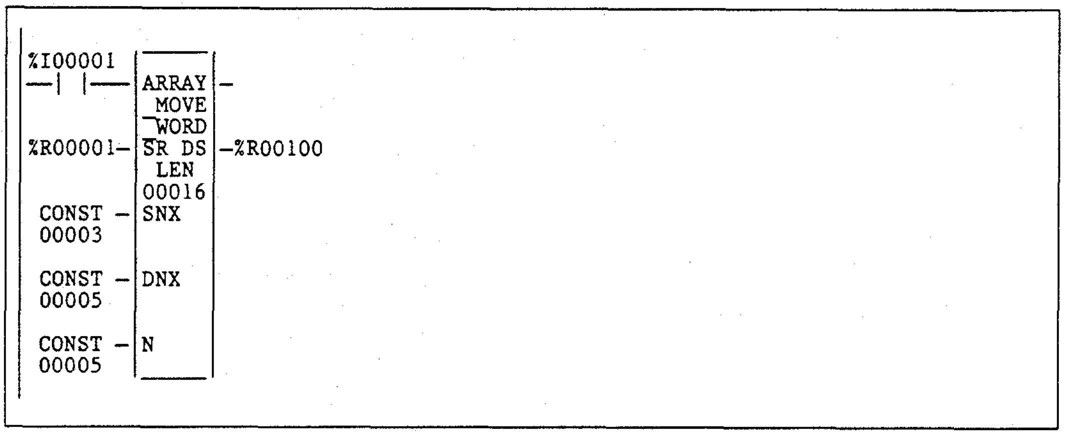
• Valid reference or place where power may flow through the function. For ARRAY_MOVE_MOVE_BIT, discrete user references %I, %Q, %M, and %T need not be aligned.
o Valid reference for INT, UINT, BIT, BYTE, or WORD data only; not valid for DINT or DWORD. %U is allowed for ARRAY_MOVE_BIT only.
Δ Valid data type for BIT, BYTE, or WORD data only; not valid for INT, UINT, DINT, or DWORD.
† %SA, %SB, %SC only; %S cannot be used.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

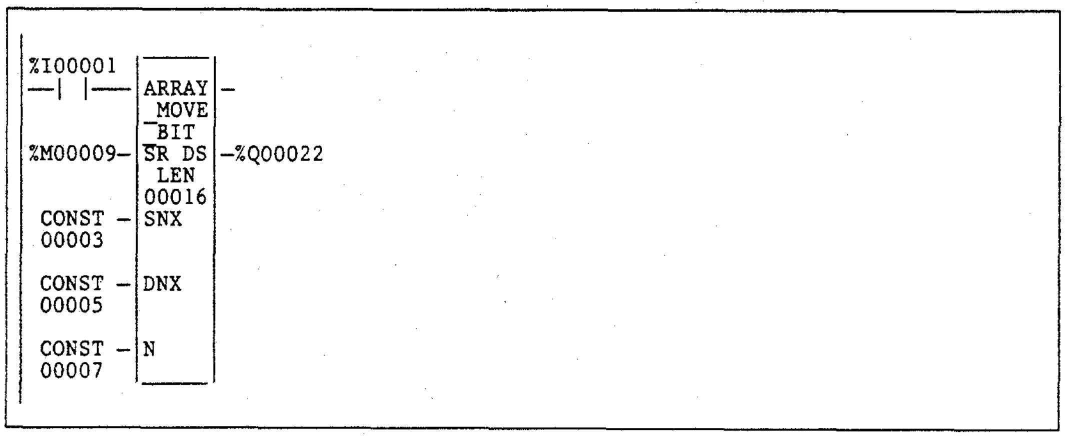
Example 1:

In this example, %R00003 - %R00007 of the array %R00001 - %R00016 is read and then written into %R00104 - %R00109 of the array %R00100 - %R00115.



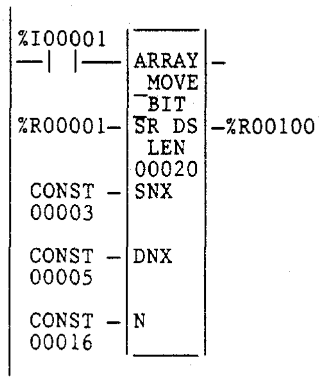
Example 2:

Using bit memory for SR and DS, %M00011 - %M00017 of the array %M00009 - %M00024 is read and then written to %Q00026 - %Q00032 of the array %Q00022 - %Q00037.



Example 3:

Using word memory, for SR and DS, the third least significant bit of %R00001 through the second least significant bit of %R00002 of the array containing all 16 bits of %R00001 and four bits of %R00002 is read and then written into the fifth least significant bit of %R00100 through the fourth least significant bit of %R00101 of the array containing all 16 bits of %R00100 and four bits of %R00101.



SRCH_EQ and SRCH_NE (INT, UINT, DINT, BYTE, WORD, DWORD) SRCH_GT and SRCH_LT SRCH_GE and SRCH_LE

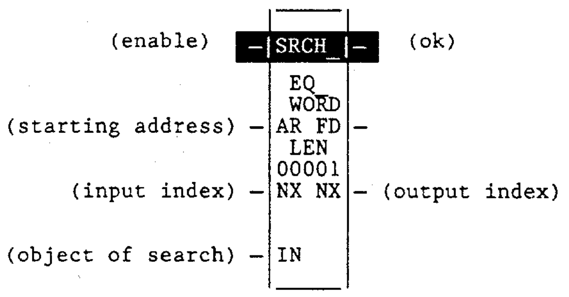
Use the appropriate Search function listed below to search for all array values for that particular operation.

Abbreviation	Function	Description
SRCH_EQ	Search Equal	Search for all array values equal to a specified value.
SRCH_NE	Search Not Equal	Search for all array values not equal to a specified value.
SRCH_GT	Search Greater Than	Search for all array values greater than a specified value.
SRCH_GE	Search Greater Than or Equal	Search for all array values greater than or equal to a specified value.
SRCH_LT	Search Less Than	Search for all array values less than a specified value.
SRCH_LE	Search Less Than or Equal	Search for all array values less than or equal to a specified value.

Each function has four input parameters and two output parameters. When the function receives power, the array is searched starting at (AR + input NX). This is the starting address of the array (AR) plus the index into this array (input NX).

The search continues until the array element of the search object (IN) is found or until the end of the array is reached. If an array element is found, output parameter (FD) is set ON and output parameter (output NX) is set to the relative position of this element within the array. If no array element is found before the end of the array is reached, then output parameter (FD) is set OFF and output parameter (output NX) is set to zero.

The valid values for input NX are 0 to LEN − 1. NX should be set to zero to begin searching at the first element. This value increments by one at the time of execution. Therefore, the values of output NX are 1 to LEN. If the value of input NX is out-of-range, (< 0 or ≥ LEN), the value of output NX is set to the default value of zero.



Note

Input and output indexes are zero-based (i.e., 0 means first reference), unlike array indexes which are one-based (i.e., 1 means first reference).

Parameters:

Parameter	Description
enable	When the function is enabled, the operation is performed.
AR	AR contains the starting address of the array to be searched.
Input NX	Input NX contains the index into the array at which to begin the search.
IN	IN contains the object of the search.
Output NX	Output NX holds the position within the array of the search target.
FD	FD indicates that an array element has been found and the function was successful.
ok	The ok output is energized when the function is performed without error. If NX is out of range, ok will not be energized.
LEN	LEN specifies the number of elements starting at AR that make up the array to be searched. It may be 1 to 32,767 bytes or words.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
AR	•	o	o	o	o	Δ†	o		•	•	•	•	•	•		
NX in	•	•	•	•	•		•		•	•	•	•	•	•	•	
IN	•	o	o	o	o	Δ†	o		•	•	•	•	•	•	•	
NX out	•	•	•	•	•		•		•	•	•	•	•	•		
FD	•															•
ok	•															•

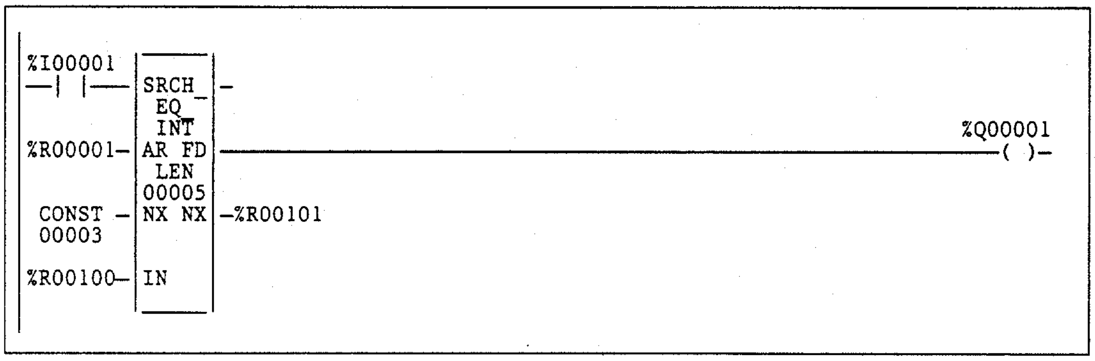
• Valid reference or place where power may flow through the function.
o Valid reference for INT, UINT, BYTE, or WORD data only; not valid for DINT or DWORD.
Δ Valid reference for BYTE,or WORD data only; not valid for INT, UINT, DINT, or DWORD.
† %SA, %SB, %SC only; %S cannot be used.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

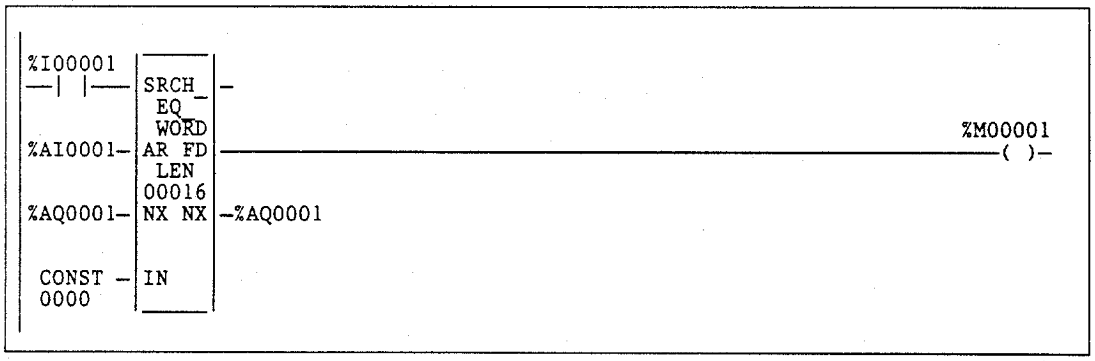
Example 1:

The array AR is defined as memory addresses %R00001 - %R00005. When EN is ON, the portion of the array between %R00004 and %R00005 is searched for an element whose value is equal to IN. If %R00001 = 7, %R00002 = 9, %R00003 = 6, %R00004 = 7, %R00005 = 7, and %R00100 = 7, then the search will begin at %R00004 and conclude at %R00004 when FD is set ON and a 4 is written to %R00101.



Example 2:

Array AR is defined as memory addresses %AI0001 - %AI0016. The values of the array elements are 100, 20, 0, 5, 90, 200, 0, 79, 102, 80, 24, 34, 987, 8, 0, and 500. Initially, %AQ0001 is 5. When EN is ON, each sweep will search the array looking for a match to the IN value of 0. The first sweep will start searching at %AI0006 and find a match at %AI0007, so FD is ON and %AQ0001 is 7. The second sweep will start searching at %AI0008 and find a match at %AI0015, so FD remains ON and %AQ0001 is 15. The next sweep will start at %AI0016. Since the end of the array is reached without a match, FD is set OFF and %AQ0001 is set to zero. The next sweep will start searching at the beginning of the array.



ARRAY RANGE (INT, DINT, WORD, DWORD)

The ARRAY RANGE function is used to compare a single input value against two arrays of delimiters that specify an upper and lower bound to determine if the input value falls within the range specified by the delimiters. The output will be an array of bits that is set ON (1) when the input value is greater than or equal to the lower limit and less than or equal to the upper limit. The output is set OFF (0) when the input is outside this range or when the range is invalid, as when the lower limit exceeds the upper limit.

Note

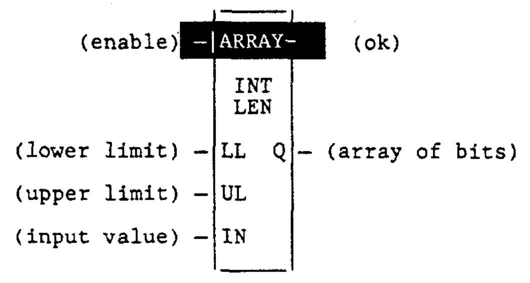
The Array Range function is only available on a Release 5 or higher CPU.

The ARRAY RANGE function operates on these types of data:

Data Type	Description
INT	Signed integer.
DINT	Double precision signed integer.
UINT	Unsigned integer.
WORD	Word data type.
DWORD	Double word data type.

The default data type is signed integer; however, it can be changed after selecting the function. For more information on data types, please refer to "Data Types" section on page 2-16.

When the function is enabled, the ARRAY RANGE function block will compare the value in input parameter IN against each range specified by the array element values of LL and UL. Output Q sets a bit ON (1) for each corresponding array element where the value of IN is greater than or equal to the value of LL and is less than or equal to the value of UL. Output Q sets a bit OFF (0) for each corresponding array element where the value of IN is not within this range or when the range is invalid, as when the value of LL exceeds the value of UL. If the operation is successful, the ok output will receive power flow.



Parameters:

Parameter	Description
enable	When the function is enabled, the operation is performed.
LL	LL contains the lower limit of the range.
UL	UL contains the upper limit of the range.
IN	IN contains the value to be compared against each range specified by L1 and L2.
ok	The ok output is energized unless an error is encountered.
Q	Output Q is energized when the value in IN is within the range specified by L1 and L2, inclusive. __Note__: Q is not aligned. It is displayed in bit format. It will display either a 1 (ON) or a 0 (OFF) for the first array element. For discrete references, it represents the reference displayed. For word references, it represents the low order bit of the reference displayed.
LEN	LEN is the number of elements in the arrays.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
LL		o	o	o	o		o		•	•	•	•	•		•‡	
UL		o	o	o	o		o		•	•	•	•	•		•‡	
IN		o	o	o	o		o		•	•	•	•	•			
ok		o	o	o	o		o		•	•	•	•	•			
Q	•															•

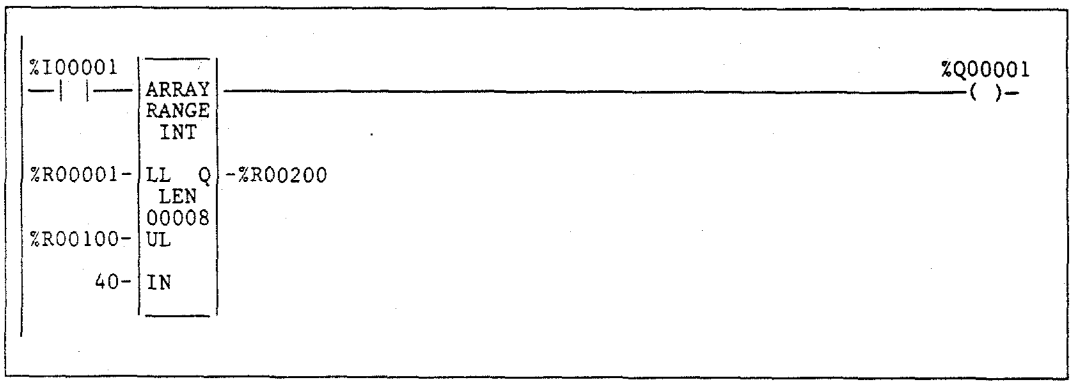
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for INT or WORD data only; not valid for DINT or DWORD.
‡ Constants are limited to integer values for double precision signed integer operations.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

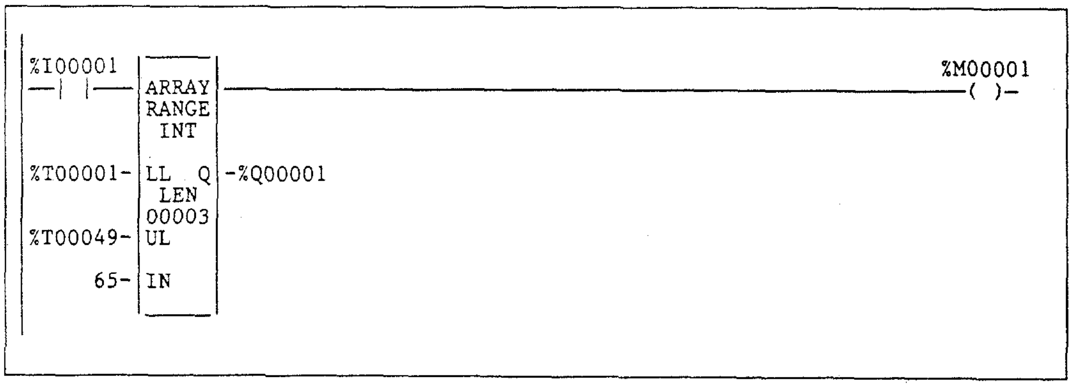
Example 1:

In the following example, the lower limit (LL) values of %R00001 through %R00008 are 1, 20, 30, 100, 25, 50, 10, and 200. The upper limit (UL) values of %R00100 through %R00108 are 40, 50, 150, 2, 45, 90, 250, and 47. The resulting Q values will be placed in the first 8 bits of %R00200. The bit values low order to high are: 1, 1, 1, 0, 1, 0, 1, and 0. The bit value displayed will be set ON (1) for the low order bit of %R00200. The ok output will be set ON (1).



Example 2:

In the following example, the lower limit (LL) array contains %T00001 through %T00016, %T00017 through %T00032, and %T00033 through %T00048. The lower limit values are 100, 65, and 1. The upper limit (UL) values are 29, 165, and 2. The resulting Q values of 0, 1, and 0 will be placed in %Q00001 through %Q00003. The bit value displayed will be 0 (OFF), representing the value of %Q00001. The ok output will be set ON (1).



Section 8: Conversion Functions

Use the conversion functions to convert a data item from one number type to another. Many programming instructions, such as math functions, must be used with data of one type. This section describes the following conversion functions:

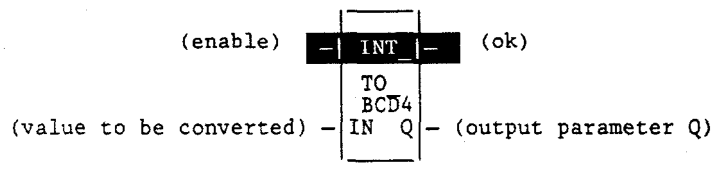
Abbreviation	Function	Description	Page
BCD-4	Convert to BCD-4	Convert an unsigned or signed integer to 4-digit BCD format.	4-151
BCD-8	Convert to BCD-8	Convert a double precision signed integer to 8-digit BCD format.	4-153
UINT	Convert to Unsigned Integer	Convert BCD-4, signed integer, or double precision signed integer to unsigned integer format.	4-155
INT	Convert to Signed Integer	Convert BCD-4, unsigned integer, or double precision signed integer to signed integer format.	4-157
DINT	Convert to Double Precision Signed Integer	Convert BCD-8, unsigned integer, or signed integer to double precision signed integer format.	4-159
REAL	Convert to Real	Convert BCD-4 BCD-8, unsigned integer, signed integer, or double precision signed integer to real value format.	4-161
TRUN	Truncate	Round the real number toward zero.	4-163

→BCD-4 (INT, UINT)

The Convert to BCD-4 function is used to output the 4-digit BCD equivalent of unsigned or signed integer data. The original data is not changed by this function. The output data can be used directly as input for another program function.

Data can be converted to BCD format to drive BCD-encoded LED displays or presets to external devices, such as high-speed counters.

When the function receives power flow, it performs the conversion, making the result available via output Q. The function passes power flow when power is received, unless the specified conversion would result in a value that is outside the range 0 to 9999.



Parameters:

Parameter	Description
enable	When the function is enabled, the conversion is performed.
IN	IN contains a reference for the integer value to be converted to BCD-4.
ok	The ok output is energized when the function is performed without error.
Q	Q contains the BCD-4 form of the original value in IN.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
IN	•	•	•	•	•		•		•	•	•	•	•	•	•	
ok	•															•
Q	•	•	•	•	•		•		•	•	•	•	•	•		

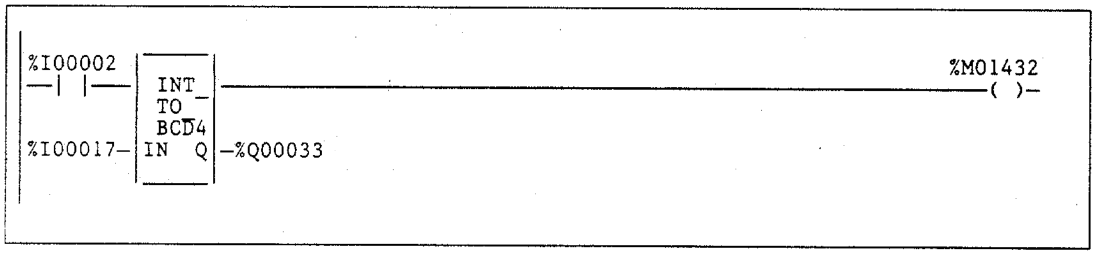
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function..

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, whenever input %I00002 is set and no errors exist, the integer at input location %I00017 through %I00032 is converted to four BCD digits and the result is stored in memory locations %Q00033 through %Q00048. Coil %M01432 is used to check for successful conversion.



→BCD-8 (DINT)

The Convert to BCD-8 function is used to output the 8-digit BCD equivalent of double precision signed integer data. The original data is not changed by this function. The output data can be used directly as input for another program function.

When the function receives power flow, it performs the conversion, making the result available via output Q. The function passes power flow when power is received, unless the specified conversion would result in a value that is outside the range 0 to 99999999.

```
                                 _______
              (enable)        -|DINT_ |-     (ok)
                               |       |
                               |  TO   |
                               | BCD8  |
(value to be converted) -|IN   Q|- (output parameter Q)
                               |_______|
```

Parameters:

Parameter	Description
enable	When the function is enabled, the conversion is performed.
IN	IN contains a reference for the integer value to be converted to BCD-8.
ok	The ok output is energized when the function is performed without error.
Q	Q contains the BCD-8 form of the original value in IN.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
IN	•								•	•	•	•	•	•	•	
ok	•															•
Q	•								•	•	•	•	•	•		

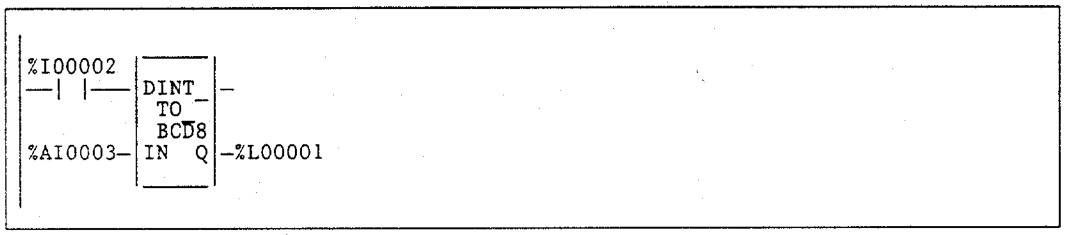
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function..

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, whenever input %I00002 is set and no errors exist, the double precision signed integer at input location %AI0003 is converted to eight BCD digits and the result is stored in memory locations %L00001 through %L00002.

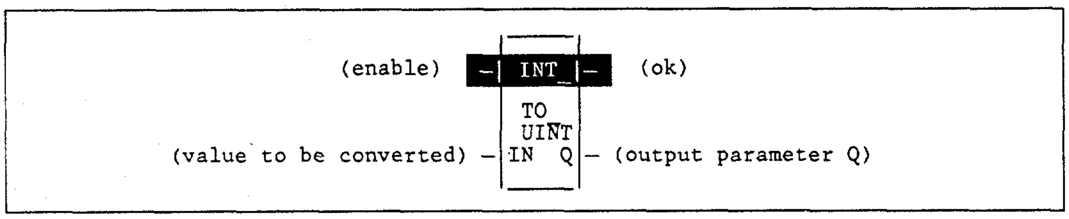


→UINT (INT, DINT, BCD-4, REAL)

The Convert to Unsigned Integer function is used to output the integer equivalent of signed integer, double precision signed integer, BCD-4, or real data. The original data is not changed by this function. The output data can be used directly as input for another program function.

One use of the →UINT function is to convert BCD data from the I/O structure into integer data and store it in memory. This can provide an interface to BCD thumbwheels or external BCD electronics, such as high-speed counters and position encoders.

When the function receives power flow, it performs the conversion, making the result available via output Q. The function passes power flow when power is received, unless the specified conversion would result in a value that is outside the range 0 to +65,535.



Parameters:

Parameter	Description
enable	When the function is enabled, the conversion is performed.
IN	IN contains a reference for the value to be converted to unsigned integer.
ok	The ok output is energized when the function is performed without error.
Q	Q contains the unsigned integer form of the original value in IN.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
IN	•	o	o	o	o		o		•	•	•	•	•	•	•	
ok	•															•
Q	•	•	•	•	•		•		•	•	•	•	•	•		

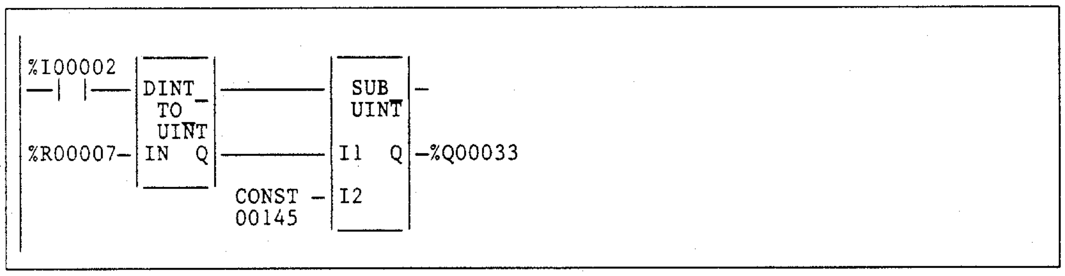
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Not valid for DINT_TO_UINT..

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, whenever input %I00002 is set and no errors exist, the double precision signed integer at input location %R00007 is converted to an unsigned integer and passed to the SUB function, where the constant value 145 is subtracted from it. The result of the subtraction is stored in the output reference location %Q00033.



→INT (UINT, DINT, BCD-4, REAL)

The Convert to Signed Integer function is used to output the integer equivalent of unsigned integer, double precision signed integer, or BCD-4 data. The original data is not changed by this function. The output data can be used directly as input for another program function.

When the function receives power flow, it performs the conversion, making the result available via output Q. The function always passes power flow when power is received, unless the data is out of range.

```
                               _________
            (enable)         -|UINT_    |-    (ok)
                              |         |
                              |   TO    |
                              |   INT   |
 (value to be converted)     -|IN      Q|-    (output parameter Q)
                              |_________|
```

Parameters:

Parameter	Description
enable	When the function is enabled, the conversion is performed.
IN	IN contains a reference for the value to be converted to integer.
ok	The ok output is energized whenever enable is energized, unless the data is out of range.
Q	Q contains the integer form of the original value in IN.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
IN	•	o	o	o	o		o		•	•	•	•	•	•	•	
ok	•															•
Q	•	•	•	•	•		•		•	•	•	•	•	•		

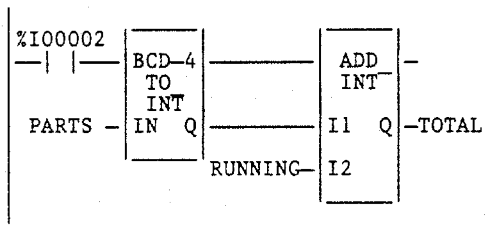
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Not valid for DINT_TO_INT..

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

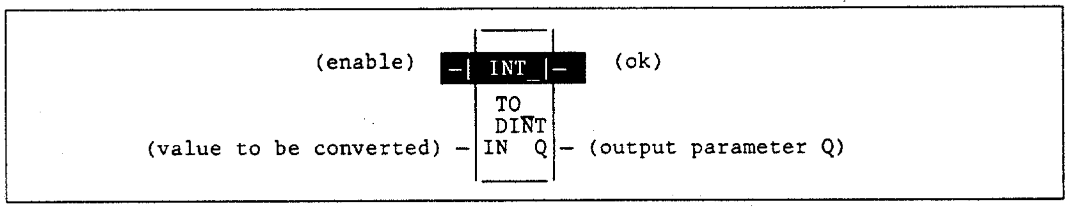
In the following example, whenever input %I00002 is set, the BCD-4 value in PARTS is converted to a signed integer and passed to the ADD function, where it is added to the signed integer value represented by the reference RUNNING. The sum is output by the ADD function to the reference TOTAL.



→DINT (INT, UINT, BCD-8, REAL)

The Convert to Double Precision Signed Integer function is used to output the double precision signed integer equivalent of unsigned integer, signed integer, or BCD-8 data. The original data is not changed by this function. The output data can be used directly as input for another program function.

When the function receives power flow, it performs the conversion, making the result available via output Q. The function always passes power flow when power is received, unless the real value is out of range.



Parameters:

Parameter	Description
enable	When the function is enabled, the conversion is performed.
IN	In contains a reference for the value to be converted to double precision integer.
ok	The ok output is energized whenever enable is energized, unless the real value is out of range.
Q	Q contains the double precision signed integer form of the original value in IN.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
IN	•	o	o	o	o		o		•	•	•	•	•	•	•	
ok	•															•
Q	•								•	•	•	•	•	•		

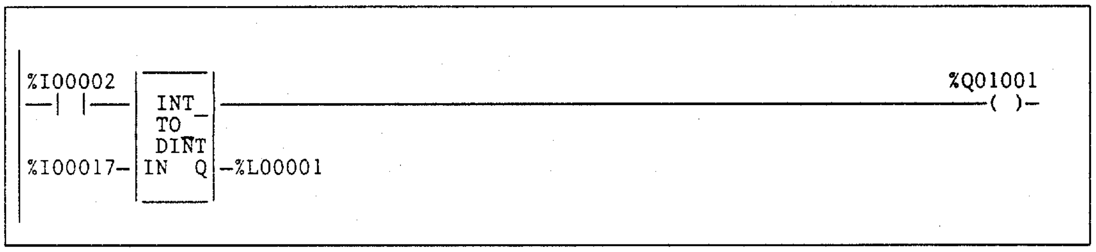
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Not valid for BCD8_TO_DINT.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, whenever input %I00002 is set, the integer value at input location %I00017 is converted to a double precision signed integer and the result is placed in location %L00001. The output %Q01001 is set whenever the function executes successfully.

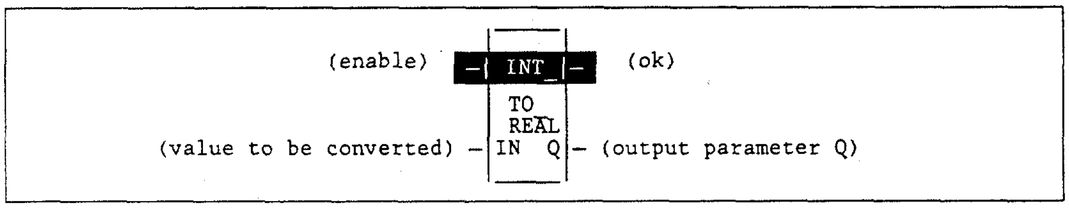


→REAL (INT, UINT, DINT, BCD-4, BCD-8)

The Convert to Real function is used to output the real value of the input data. The original data is not changed by this function. The output data can be used directly as input for another program function.

When the function receives power flow, it performs the conversion, making the result available via output Q. The function passes power flow when power is received, unless the specified conversion would result in a value that is out of range.

It is possible for a loss of precision to occur when converting from DINT or BCD-8 to REAL since the number of significant bits is reduced to 24.



Parameters:

Parameter	Description
enable	When the function is enabled, the conversion is performed.
IN	IN contains a reference for the integer value to be converted to REAL.
ok	The ok output is energized when the function is performed without error.
Q	Q contains the REAL form of the original value in IN.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
IN	•	o	o	o	o		o		•	•	•	•	•	•	•	
ok	•															•
Q	•								•	•	•	•	•	•		

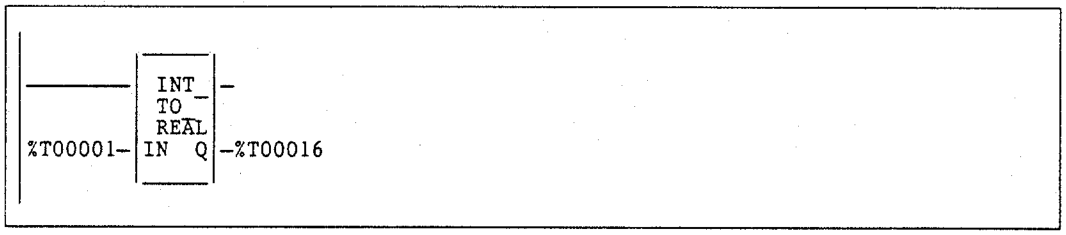
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Not valid for DINT._TO_REAL or BCD8_TO_REAL..

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

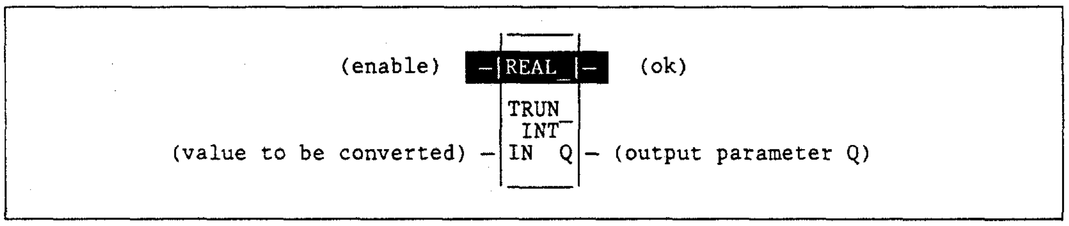
In the following example, the integer value of input IN is 678. The result value placed in %T00016 is 678.000.



TRUN (INT, DINT)

The Truncate function is used to round the real number toward zero. The original data is not changed by this function. The output data can be used directly as input for another program function.

When the function receives power flow, it performs the conversion, making the result available via output Q. The function passes power flow when power is received, unless the specified conversion would result in a value that is out of range or unless IN is NaN (Not a Number).



Parameters:

Parameter	Description
enable	When the function is enabled, the conversion is performed.
IN	IN contains a reference for the real value to be truncated.
ok	The ok output is energized when the function is performed without error, unless the value is out of range or IN is NaN.
Q	Q contains the truncated INT or DINT value of the original value in IN.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
IN	•								•	•	•	•	•	•	•	
ok	•															•
Q	•	o	o	o	o		o		•	•	•	•	•	•		

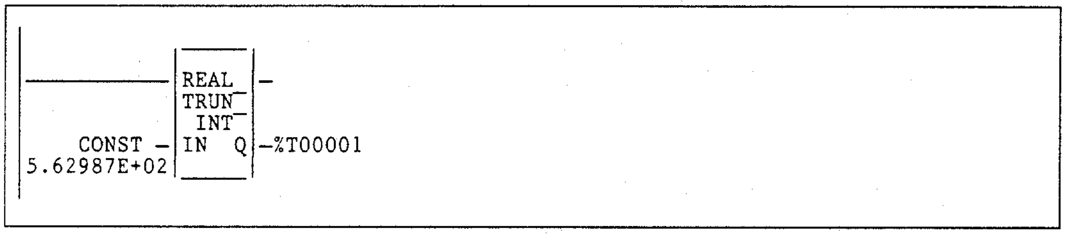
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Not valid for REAL_TRUN_INT only..

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, the displayed constant is truncated and the integer result 562 is placed in %T00001.



Section 9: Control Functions

This section describes the control functions, which may be used to limit program execution and change the way the CPU executes the application program. (Refer to chapter 2, section 1, "PLC Sweep Summary," for information on the CPU sweep.)

Function	Description	Page
CALL	Causes program execution to go to a specified program block.	4-167
CALL EXTERNAL	Causes program execution to go to a specified external block.	4-168
CALL SUBROUTINE	Causes program execution to go to a specified parameterized subroutine block.	4-170
DOIO	Services for one sweep a specified range of inputs or outputs immediately. (All inputs or outputs on a module are serviced if any reference locations on that module are included in the DO I/O function. Partial I/O module updates are not performed.) Optionally, a copy of the scanned I/O can be placed in internal memory, rather than the real input points.	4-174
SUSIO	Suspends for one sweep all normal I/O updates, except those specified by DO I/O instructions.	4-178
MCR	Programs a Master Control Relay. An MCR causes all rungs between the MCR and its subsequent ENDMCR to be executed without power flow.	4-180
ENDMCR	Indicates that the subsequent logic is to be executed with normal power flow.	4-181
JUMP	Causes program execution to jump to a specified location (indicated by a LABEL, see below) in the logic.	4-182
LABEL	Specifies the target location of a JUMP instruction.	4-183
COMMENT	Places a comment (rung explanation) in the program. After entering the instruction, the text can be typed in by zooming into the instruction.	4-184
FOR, END_FOR, EXIT	Repeat logic a specified number of times within a program.	4-185

Function	Description	Page
SVCREQ	Requests one of the following special PLC services:	4-189
	Change/Read Constant Sweep Timer.	4-191
	Read Window Values.	4-194
	Change Programmer Communications Window State and Values.	4-195
	Change System Communications Window State and Values.	4-196
	Change/Read Checksum Task State & No. of Words to Checksum.	4-200
	Change/Read Time-of-Day Clock State and Values.	4-202
	Reset Watchdog Timer.	4-206
	Read Sweep Time from Beginning of Sweep.	4-207
	Read Program Name This Block is In.	4-208
	Read PLC ID.	4-209
	Read PLC Run State.	4-210
	Shut Down PLC.	4-211
	Clear Fault Tables.	4-212
	Read Last-Logged Fault Table Entry.	4-213
	Read Elapsed Time Clock.	4-217
	Mask/Unmask I/O Interrupt.	4-218
	Read I/O Override Status.	4-220
	Set Run Enable/Disable.	4-221
	Read Fault Tables.	4-222
	Log User-Defined PLC Fault.	4-225
	Mask/Unmask Timed Interrupts.	4-228
	Read Master Checksum.	4-229
	Disable/Enable EXE Block Checksum.	4-231
	Role Switch.	4-232
	Write to Reverse Transfer Area.	4-233
	Read from Reverse Transfer Area.	4-233
	Suspend/Resume I/O Interrupt	4-235
PID	Provides two PID (proportional/integral/derivative) closed-loop control algorithms: Standard ISA PID algorithm (PIDISA) and Independent term algorithm (PIDIND).	4-237

CALL

Use the CALL function to cause program execution to go to a specified program block. A CALL function can be used in any _MAIN program block, program block, or Parameterized Subroutine Block .

Note

The maximum number of program block calls that can be programmed in a program block is 64. The maximum number of declared program blocks is 255. The maximum number of times you can call a subroutine is 255.

When the CALL function receives power flow, it causes the scan to go immediately to the designated program block and execute it. After the program block execution is complete, control returns to the point in the logic immediately following the CALL instruction.

CALL EXTERNAL

Use the CALL EXTERNAL function to cause program execution to go to a specified external block created outside the Logicmaster software. The eight CALL EXTERNAL functions vary in the number of parameters which convey input and output data blocks. The input and output parameters may have a length of one bit or one word. When a contact instruction adjoins one of the input or output parameters, the parameter is one bit in size. All data flow will be one bit in length. The enable and ok parameters are present for each function.

When the function is enabled, the specified external block is executed. The external logic operates on the blocks of incoming data and produces the blocks of output data. The ok parameter is controlled by logic within the external block.

After the program block execution is complete, control returns to the point in the logic immediately following the CALL instruction.

The CALL EXTERNAL function with no input and output parameters has the following form:

```
                 ______________
                |              |
(enable)       -|CALL blkname  |-     (ok)
                |  (EXTERNAL)  |
                |______________|
```

The CALL EXTERNAL function may have up to seven pairs of parameters, as shown below:

```
                              ______________
                             |              |
              (enable)      -|CALL blkname  |-     (ok)
                             |  (EXTERNAL)  |
(input parameter I1) -|IN1          Q1|- (output parameter Q1)
                             |              |
(input parameter I2) -|IN2          Q2|- (output parameter Q2)
                             |              |
(input parameter I3) -|IN3          Q3|- (output parameter Q3)
                             |              |
(input parameter I4) -|IN4          Q4|- (output parameter Q4)
                             |              |
(input parameter I5) -|IN5          Q5|- (output parameter Q5)
                             |              |
(input parameter I6) -|IN6          Q6|- (output parameter Q6)
                             |              |
(input parameter I7) -|IN7          Q7|- (output parameter Q7)
                             |______________|
```

Parameters:

Parameter	Description
enable	When the function is enabled, the operation is performed.
IN1-IN7	The input parameters IN1 through IN7 indicate the starting location of data to be used within the external block.
ok	The ok output is controlled by logic within the external block.
Q1-Q7	The output parameters Q1 through Q7 indicate the starting location of data with output values.

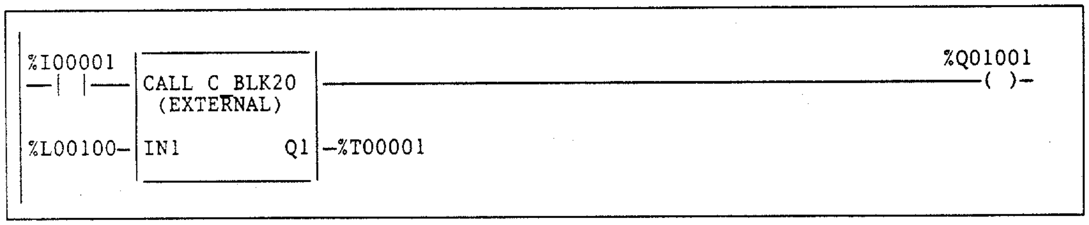
Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
IN1-IN7	•	•	•	•		•	•	•	•	•	•	•	•	•	•	
ok	•															
Q1-Q7	•	•	•	•		•	•	•	•	•	•	•		•	•	

• Valid reference or place where power may flow through the function.

Example:

In the following example, if %I00001 is set, the external block named C_BLK20 is executed. Block C_BLK20 operates on the input data located at reference address %L00100 and produces values in the block of output data located at reference address %T00001. Logic within C_BLK20 controls the Q1 output.



CALL SUBROUTINE

Use the CALL SUBROUTINE function to cause program execution to go to a specified parameterized subroutine. Parameterized subroutine blocks (PSBs) are user-defined function blocks, configured with between zero and seven input/output parameter pairs. A program can "call" a Parameterized Subroutine Block as it executes; however, the Parameterized Subroutine Block must be declared through the block declaration editor before a CALL instruction can be used for that Parameterized Subroutine Block. For more information on declaring a Parameterized Subroutine Block, refer to chapter 2, section 3, "Program Organization and User Data," in this manual and the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263.

When the function is enabled, the specified parameterized subroutine is executed. The logic in the subroutine operates on the blocks of incoming data and produces blocks of output data. The ok parameter may be controlled by logic within the subroutine. After the program block execution is complete, control returns to the point in the logic immediately following the CALL instruction.

The CALL SUBROUTINE function with no input or output parameters has this form:

```
                 _____________
                |             |
   (enable)   - |CALL blkname | -   (ok)
                |(SUBROUTINE) |
                |_____________|
```

In this case of a zero-parameter PSB call, the ok parameter is always set to ON.

The CALL SUBROUTINE function may have up to seven pairs of parameters, as shown below:

```
                                    _____________
                                   |             |
                       (enable)  - |CALL blkname | -  (ok)
                                   |(SUBROUTINE) |
           (input parameter X1) -  |X1         Y1| - (output parameter Y1)

           (input parameter X2) -  |X2         Y2| - (output parameter Y2)

           (input parameter X3) -  |X3         Y3| - (output parameter Y3)

           (input parameter X4) -  |X4         Y4| - (output parameter Y4)

           (input parameter X5) -  |X5         Y5| - (output parameter Y5)

           (input parameter X6) -  |X6         Y6| - (output parameter Y6)

           (input parameter X7) -  |X7         Y7| - (output parameter Y7)
                                   |_____________|
```

Parameters:

Parameter	Description
enable	When the function is enabled, the operation is performed.
X1-X7	The input parameters X1 through X7 indicate the starting location of data to be used within the subroutine block.
ok	The ok output may be controlled by logic within the subroutine block using PSB formal parameter Y0[001].
Y1-Y7	The output parameters Y1 through Y7 indicate the starting location of the output data.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
X1-X7	•†	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
ok	•															
Y1-Y7	•†	•	•	•	•		•	•	•	•	•	•	•	•		•

• Valid reference or place where power may flow through the function.
† Bit parameters only (when bit length is 1).
‡ When word length is 1.

Note

PSB (Parameterized Subroutine Block) formal BIT parameters are **not** allowed as parameters to the CALL SUBROUTINE function. PSB formal WORD and NWORD parameters are allowed as parameters to the CALL SUBROUTINE function. For restrictions within a Parameterized Subroutine Block, see page 2-34.

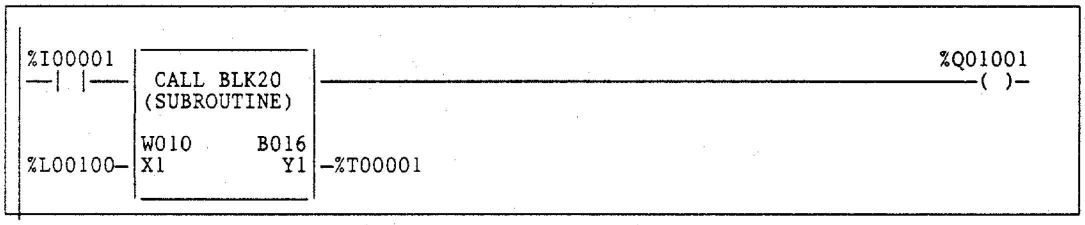
Note

Indirect references may be allowed as assigned input parameters, but not as formal parameters. %S references, like ALW_ON and ALW_OFF, may be required in some applications to take the place of required but unused assigned input parameters. %S references may not be assigned to WORD type input parameters, but may be assigned to BIT type input parameters.

Also note that a discrete reference (e.g., %I, %Q, %M, %T, %S,) cannot be assigned to an NWORD parameter but can be assigned to a WORD parameter.

Example:

In the following example, if %I00001 is set, the Parameterized Subroutine Block named BLK20 is executed. Block BLK20 operates on the input data located at reference addresses %L00100 – %L00109 and produces values in the block of output data located at reference addresses %T00001 – %T00016. Logic within BLK20 controls the Y1 output.



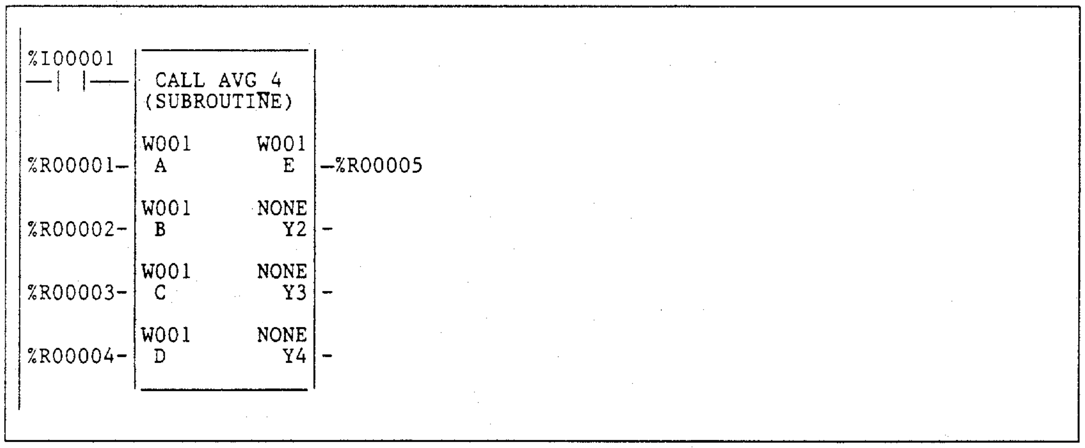
Parameterized Subroutine Block Example

Parameterized subroutine blocks are useful for building libraries of user-defined functions that can be used and moved from one application to another. For example, if you had a mathematical equation or algorithm that needed to be performed several times, you could write a Parameterized Subroutine Block for that function and then call it as needed throughout your program.

For example, if you have a mathematical equation such as:

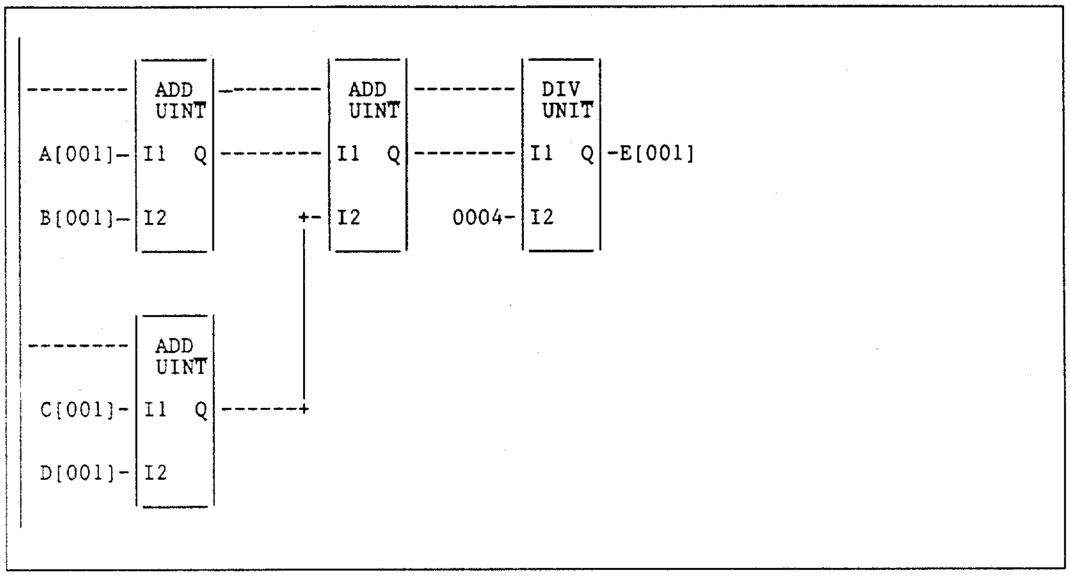
$$E = (A + B + C + D) / 4$$

where E is a word output, and A, B, C, and D are word inputs. A parameterized subroutine named AVG_4 could be defined and called as follows:



In this example, the average of %R00001, %R00002, %R00003, and %R00004 would be placed in %R00005.

The logic within the Parameterized Subroutine Block would then look like this:



DOIO

The DO I/O (DOIO) function is used to update inputs or outputs for one scan while the program is running. The DOIO function can be used in conjunction with a SUSIO function, which stops the normal I/O scan. It can also be used to update selected I/O during the program in addition to the normal I/O scan.

If input references are specified, the function allows the most recent values of inputs to be obtained for program logic. If output references are specified, DO I/O updates outputs based on the most current values stored in I/O memory. I/O points are serviced in increments of entire I/O modules; the PLC adjusts the references, if necessary, while the function executes. The DOIO function will not scan I/O modules that are not configured.

Note

The DOIO function is supported for Series 90-70 I/O modules only. It does not support Genius I/O modules.

The DOIO function has four input parameters and one output parameter. When the function receives power flow and input references are specified, the input points at the starting reference ST and ending at END are scanned. If a reference is specified for ALT, a copy of the new input values is placed in memory, beginning at that reference, and the real input points are not updated. ALT must be the same size as the reference type scanned. If a discrete reference is used for ST and END, then ALT must also be discrete. If no reference is specified for ALT, the real input points are updated.

When the DOIO function receives power flow and output references are specified, the output points at the starting reference ST and ending at END are written to the output modules. If outputs should be written to the output modules from internal memory, other than %Q or %AQ, the beginning reference can be specified for ALT. The range of outputs written to the output modules is specified by the starting reference ST and the ending reference END.

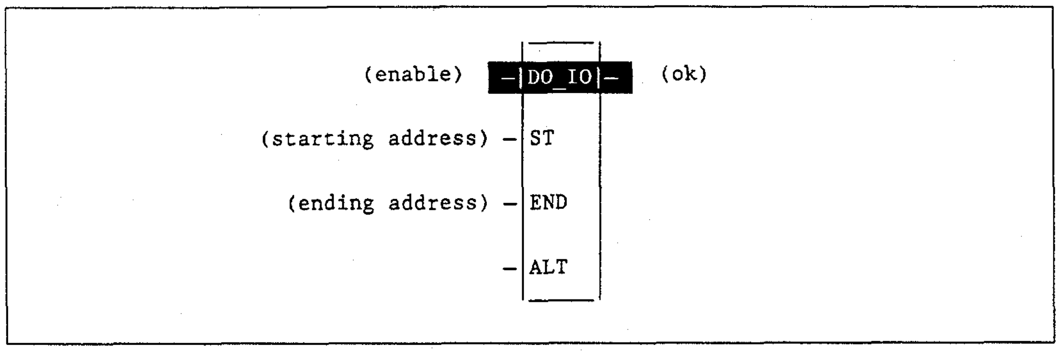
Execution of the function continues until either all inputs in the selected range have reported, or all outputs have been serviced on the I/O cards. Program execution then returns to the next function following the DO I/O.

The function passes power to the right whenever power is received, unless:

- Not all references of the type specified are present within the selected range.
- The CPU is not able to properly handle the temporary list of I/O created by the function.
- The range specified includes I/O modules that are associated with a "Loss of I/O Module" fault.

Note

If the DOIO function is used with timed or I/O interrupts, transitional contacts associated with scanned inputs may not operate as expected.



Parameters:

Parameter	Description
enable	When the function is enabled, a limited input or output scan is performed.
ST	ST is the starting address of the set of input or output points or words to be serviced.
END	END is the ending address of the set of input or output points or words to be serviced.
ALT	For the input scan, ALT specifies the address to store scanned input point/word values. For the output scan, ALT specifies the address to get output point/word values from to send to the I/O modules.
ok	The ok output is energized when the input or output scan completes normally.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
ST		•	•									•	•	•		
END		•	•									•	•	•		
ALT		•	•	•	•		•		•			•	•	•		•
ok	•															•

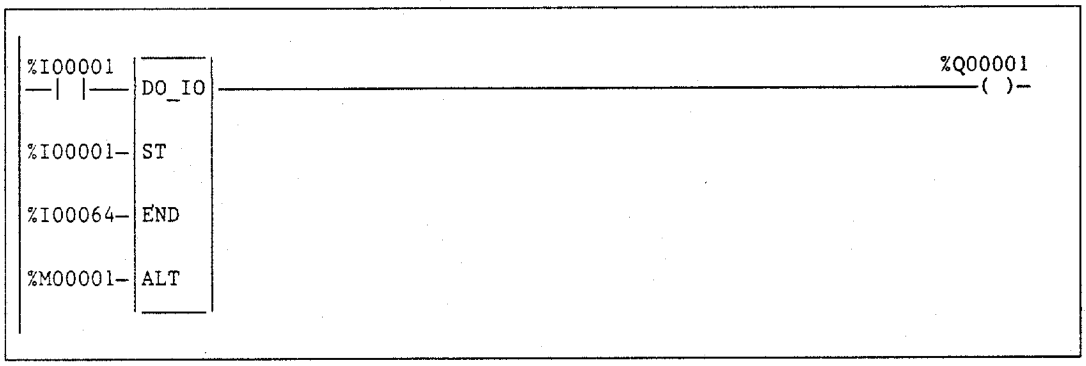
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.
o Valid reference for INT, UINT, or WORD data only; not valid for DINT or DWORD.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

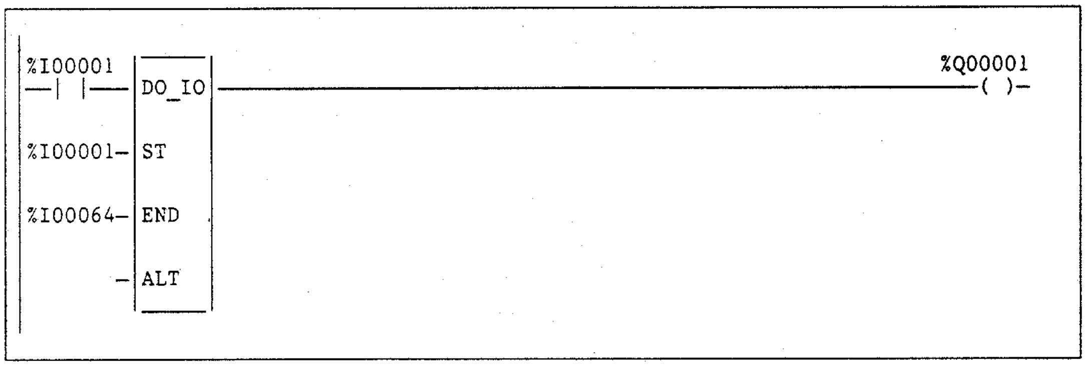
Input Example 1:

In the following example, when the enabling input %I00001 is ON, references %I00001 through %I00064 are scanned and %Q00001 is turned on. A copy of the scanned inputs is placed in internal memory from reference %M00001 through %M00064. The real input points are not updated. This form of the function can be used to compare the current values of input points with the values of input points at the beginning of the scan.



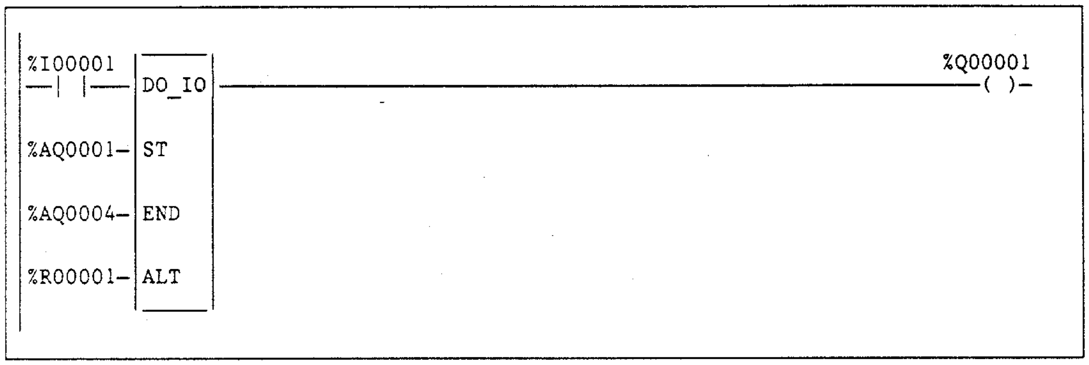
Input Example 2:

In the following example, when the enabling input %I00001 is ON, references %I00001 through %I00064 are scanned and %Q00001 is turned on. The scanned inputs are placed in the input status memory from reference %I00001 to %I00064. This form of the function allows input points to be scanned one or more times during the program execution portion of the CPU sweep.



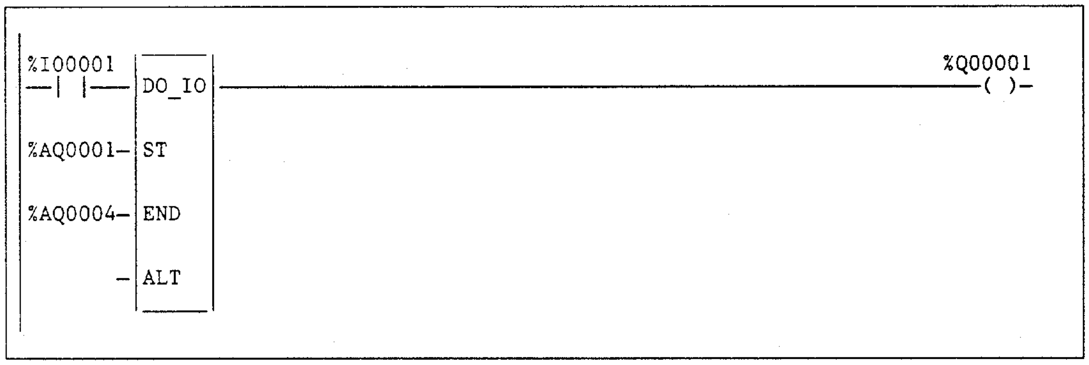
Output Example 1:

In the following example, when the enabling input %I00001 is ON, the values at references %R00001 through %R00004 are written to analog output channels %AQ0001 through %AQ0004 and %Q00001 is turned on. The values at %AQ0001 through %AQ0004 are not written to the analog output modules.



Output Example 2:

In the following example, when the enabling input %I00001 is ON, the values at references %AQ0001 through %AQ0004 are written to analog output channels %AQ0001 through %AQ0004 and %Q00001 is turned on.



SUSIO

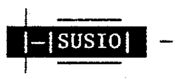
The Suspend I/O (SUSIO) function is used to stop normal I/O scans from occurring for one CPU sweep. During the next output scan, all outputs are held at their current states. During the next input scan, the input references are not updated with data from inputs. However, during the input scan portion of the sweep the CPU will verify that Genius bus controllers have completed their previous output updates.

Note

The SUSIO function suspends all I/O, both analog and discrete, whether integrated I/O or Genius I/O.

The SUSIO function has one input and one output. When the function receives power flow, all I/O servicing stops except that provided by DOIO functions. If the function were placed at the left rail of the ladder, without enabling logic to regulate its execution, no regular I/O scan would ever be performed.

The SUSIO function passes power flow to the right whenever power is received.



Example:

This example shows a SUSIO function and a DOIO function used to stop I/O scans, then cause certain I/O to be scanned from the program.

Inputs %I00010 and %I00011 form a latch circuit with the contact from %M00001. This keeps the SUSIO function active on each sweep until %I00011 goes on. If this input were not scanned by the DO after the SUSIO went active, the SUSIO could only be disabled by powering down the PLC.

Output %Q00002 is set when both DOIO functions execute successfully. The rung is constructed so that both DOIO functions will execute even if one does not set its OK output. With normal I/O suspended, output %Q00002 is not updated until a DOIO function with %Q00002 in its range executes. This will not occur until the sweep after the setting of %Q00002. Outputs that are set after a DOIO function executes are not updated until another DOIO function executes, typically in the next sweep. Because of this delay, most programs that use SUSIO and DOIO place the SUSIO function in the first rung of the program, the DOIO function that processes inputs in the next rung, and the DOIO function that processes outputs in the last rung.

The range of the DOIO function doing outputs is %Q00001 through %Q00030. If the module in this range were a 32-point module, the DOIO function would actually perform a scan of the entire module. A DOIO function will not break the scan in the middle of an I/O module.

```
|
| [ START OF PROGRAM ]
|
|%I00010 %I00011   _____                                          %M00001
|——| |——+——|/|——|SUSIO|——————————————————————————————————————————————( )—
|        |         ‾‾‾‾‾
|%M00001 |
|——| |——+
|
|%M00001  |‾‾‾‾‾‾|                                                 %M00551
|——| |————|DO_IO |————————————————————————————————————————————————————( )—
|         |      |
|%I00001——|ST    |
|         |      |
|%I00016——|END   |
|         |      |
|         |ALT   |
|         |______|
|
•
•
•
|
|%M00001  |‾‾‾‾‾‾| %M00551                                         %Q00002
|——| |————|DO_IO |————| |——————————————————————————————————————————————( )—
|         |      |
|%Q00001——|ST    |
|         |      |
|%Q00030——|END   |
|         |      |
|         |ALT   |
|         |______|
|
| [ END OF PROGRAM ]
|
```

MCR

All rungs between an active Master Control Relay (MCR) and its corresponding End Master Control Relay (ENDMCR) are executed with negative logic. Use the MCR function to cause a portion of the program logic to be bypassed. An ENDMCR function associated with the MCR is used to resume normal program execution. Unlike the JUMP instruction, MCRs can only occur in the forward direction. The ENDMCR instruction must appear after its corresponding MCR instruction in a program.

Note

> Program block calls within the scope of an active MCR will not execute. However, any timers in the program block will continue to accumulate time.

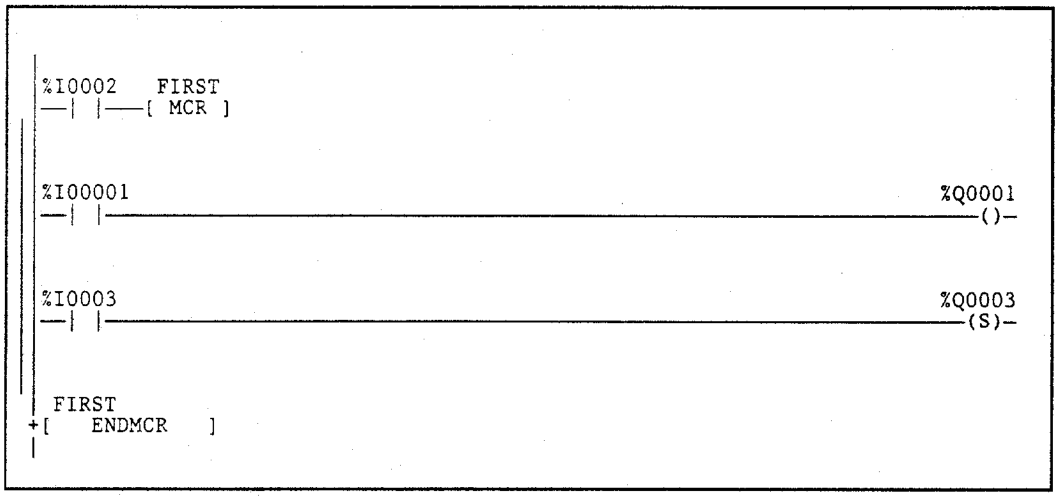
There can be only one MCR instruction for each ENDMCR instruction. An MCR/ENDMCR pair can be nested within other MCR/ENDMCR pairs.

The MCR function has an enable boolean input (EN) and also a name which identifies the MCR. This name is used again with an ENDMCR instruction. The MCR function has no outputs; there can be nothing after an MCR in a rung.

```
-[ MCR ]-
```

Example:

In the following example, whenever %I00002 allows power flow into the MCR function, program execution will continue without power flow to the coils until the associated ENDMCR is reached. If %I00001 and %I00003 are ON, %Q00001 is turned OFF and %Q00003 remains ON.



ENDMCR

Use the End Master Control Relay (ENDMCR) function to resume normal program execution after an MCR function. When the MCR associated with the ENDMCR is active, the ENDMCR causes program execution to resume with normal power flow. When the MCR associated with the ENDMCR is not active, the ENDMCR has no effect.

The ENDMCR function has a negated boolean input (EN). The instruction enable must be provided by the power rail; execution cannot be conditional. The ENDMCR function also has a name, which identifies the ENDMCR and associates it with the corresponding MCR(s). The ENDMCR function has no outputs; there can be nothing before or after an ENDMCR instruction in a rung.

```
[   END_MCR ]-
```

Example:

In the following example, an ENDMCR instruction is programmed to terminate MCR range "clear."

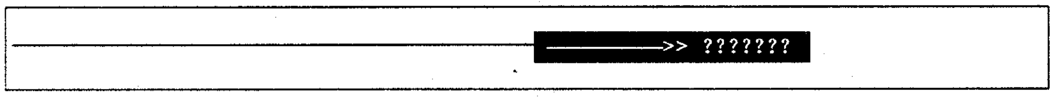
```
|
|  CLEAR
|-[       END MCR ]
|
```

JUMP

Use the JUMP instruction to cause a portion of the program logic to be bypassed. The JUMP can be either a forward or a backward JUMP. Program execution will continue at the LABEL specified.

The JUMP instruction is similar to an MCR, except that coils within the range of the JUMP are not executed with negative logic. When the JUMP is active, all coils within its scope are frozen. This includes coils associated with timers, counters, latches, and relays.

The JUMP instruction is always placed in columns 9 and 10 of the current rung line; there can be nothing after the JUMP instruction in the rung. Power flow jumps directly from the instruction to the rung with the named LABEL.

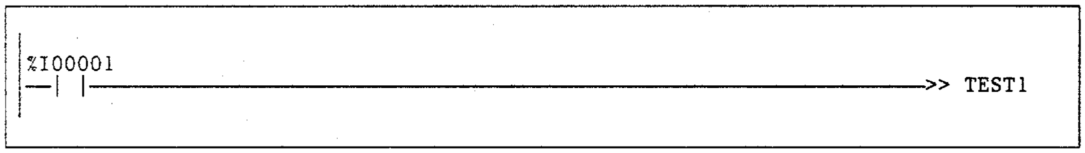


Caution

To avoid creating an endless loop with forward and backward JUMP instructions, a backward JUMP must contain a way to make it conditional.

Example:

In the following example, whenever JUMP TEST1 is active, power flow is transferred to LABEL TEST1.



LABEL

The LABEL instruction functions as the target destination of a JUMP. Use the LABEL instruction to resume normal program execution after a JUMP instruction.

There can be only one LABEL with a particular label name in a program. Programs without a matched JUMP/LABEL pair can be created and stored to the PLC, but cannot be executed.

The LABEL instruction has no input parameters and no output parameters; there can be nothing either before or after a LABEL in a rung.

```
???????
```

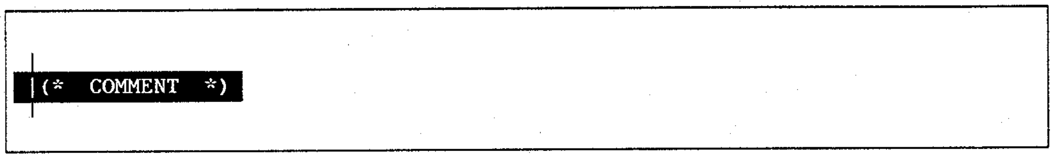
Example:

In the following example, power flow from JUMP TEST1 is resumed, starting at LABEL TEST1.

```
|
| TEST1 :
|
```

COMMENT

Use the COMMENT function to enter a comment (rung explanation) in the program. A comment can have up to 2048 characters of text. It is represented in the ladder logic like this:



The text can be read or edited by moving the cursor to **`(* COMMENT *)`** after accepting the rung and selecting Zoom (F10). Comment text can also be printed.

Longer text can be included in printouts using an annotation text file, as described below:

1. Create the comment:

   A. Enter text to the point where the text from the other file should begin.

   B. Move the cursor to the beginning of a new line and enter **`\I`** or **`\i`**, the drive followed by a colon, the subdirectory or folder, and the file name, as shown in this example:

   ```
   \I d:\text\commnt1
   ```

   If the file is located on the same drive as the program folder, the drive designation is not necessary.

   C. Continue editing the program, or exit to MS-DOS.

2. After exiting the programmer, create a text file using any MS-DOS compatible software package. Give the file the file name entered in the comment, and place it on the drive specified in the comment.

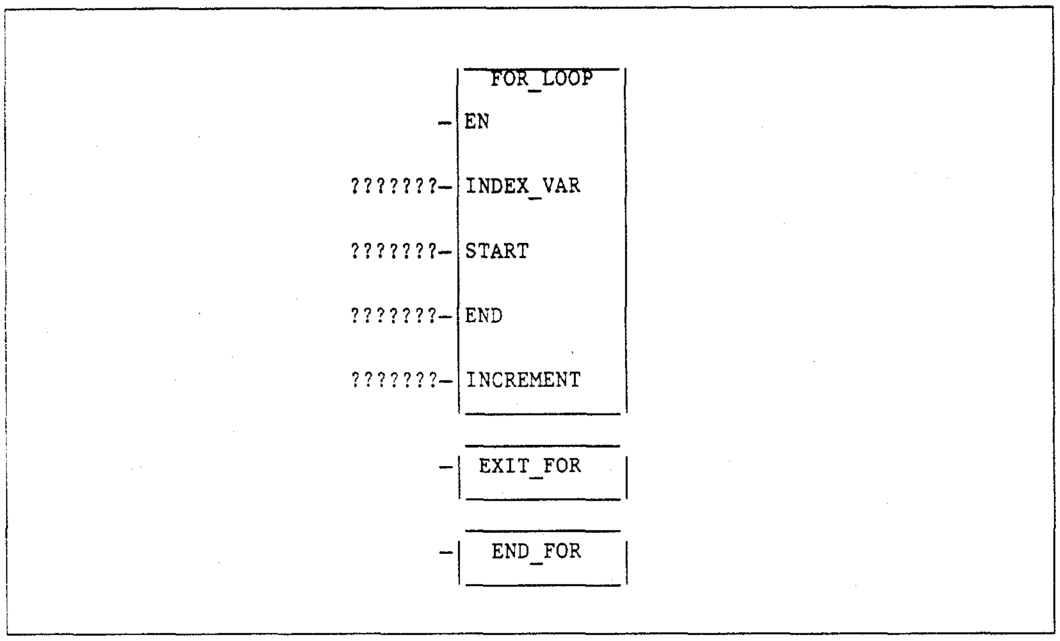
FOR, END_FOR, and EXIT

FOR, END_FOR, and EXIT instructions enable you to repeat rung logic a specified number of times while varying the value of the FOR INDEX_VAR in the loop. The rung logic to be repeated must be placed between the FOR and END_FOR instructions.

The FOR instruction has five input parameters; there are no out parameters. When there is power flow at enable EN of the FOR instruction, the START, END, and INCREMENT parameters are saved. These saved values are used to evaluate the number of times the rungs between the FOR and its END_FOR instructions are executed. Changing the START and END parameters while the FOR loop is executing will not change its operation.

When there is power flow into the EXIT instruction, the FOR instruction is terminated and power flow jumps directly to the statement following the END_FOR instruction.

There can be nothing after the FOR instruction in the rung. An EXIT statement can only be placed between a FOR instruction and an END_FOR instruction in a program, and there can be nothing after the EXIT statement in the rung, as well. The END_FOR statement must be the only instruction in the rung.



A FOR_LOOP can assign decreasing values to its index variable by setting the increment to a negative number. If the START value is 21, the END value is 1, and the increment value is −5, the statements of the FOR loop are executed five times, and the index variable is decremented by 5 in each pass. The values of the index variable will be 21, 16, 11, 6, and 1.

When the START and END values are set equal, the statements of the FOR loop are executed only once.

When START cannot be incremented to reach the END or START cannot be decremented to reach the END, the statements within the FOR loop are not executed. For example, the value of START is 10, the value of END is 5, and the INCREMENT is 1. Power flow will jump directly from the FOR statement to the statement after the END_FOR statement.

Note

If the EN parameter for the FOR_LOOP instruction has power flow when it is first tested, the rungs between the FOR and its corresponding END_FOR statement are executed the number of times initially specified by START, END, and INCREMENT. This repeated execution will occur on a single sweep of the PLC and may cause the watchdog timer to expire if the loop is long.

Nesting of FOR loops is allowed, but it is restricted to five FOR/END_FOR pairs. Each FOR instruction must have a matching END_FOR statement following it.

Nesting with JUMPs and MCRs is allowed, provided that they are properly nested. MCRs and ENDMCRs must be completely within or completely outside the scope of a FOR/END_FOR pair. JUMPs and LABEL instructions must also be completely within or completely outside the scope of a FOR/END_FOR pair. Jumping into or out of the scope of a FOR/END_FOR is not allowed.

Parameters:

Parameter	Description
enable	When the function is enabled, the operation is performed.
INDEX_VAR	INDEX_VAR contains the index variable. When the loop has completed, this value is undefined. Changing the value of the index variable within the scope of the FOR loop is not recommended.
START	START contains the index start value.
END	END contains the index end value.
INCREMENT	INCREMENT contains the increment value.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
INDEX		•	•	•	•		•		•	•	•	•	•	•		
START	•	•	•	•	•		•		•	•	•	•	•	•	•	
END	•	•	•	•	•		•		•	•	•	•	•	•	•	
INCREMENT															•	•

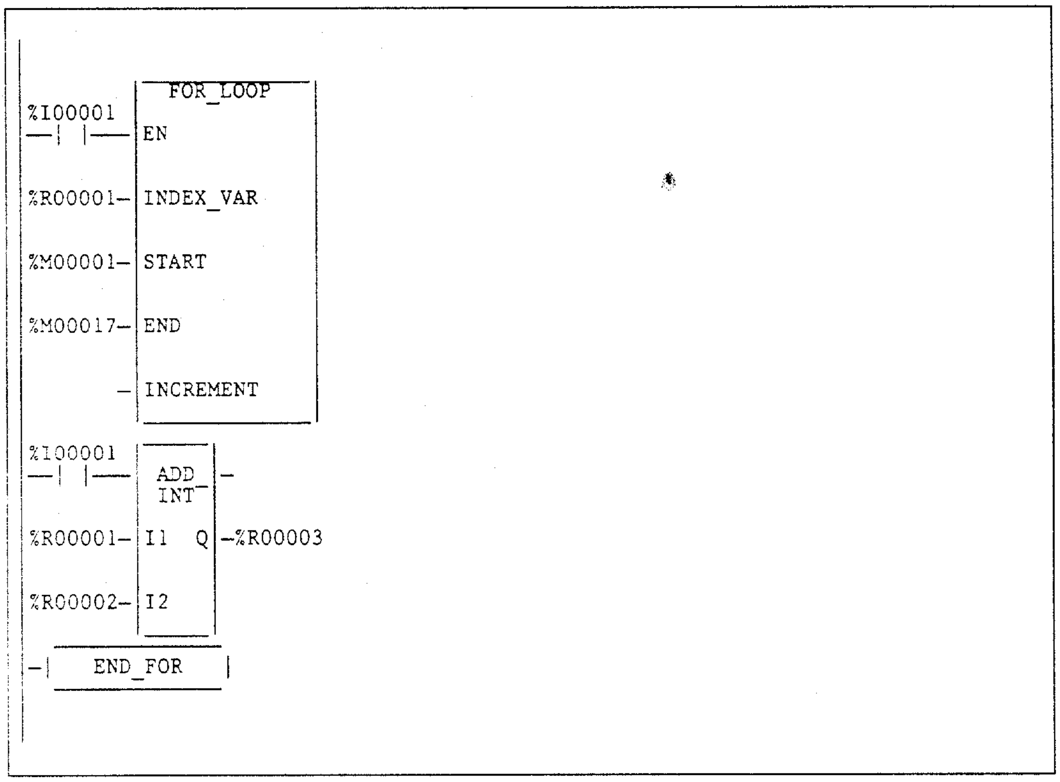
• Valid reference or place where power may flow through the function.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

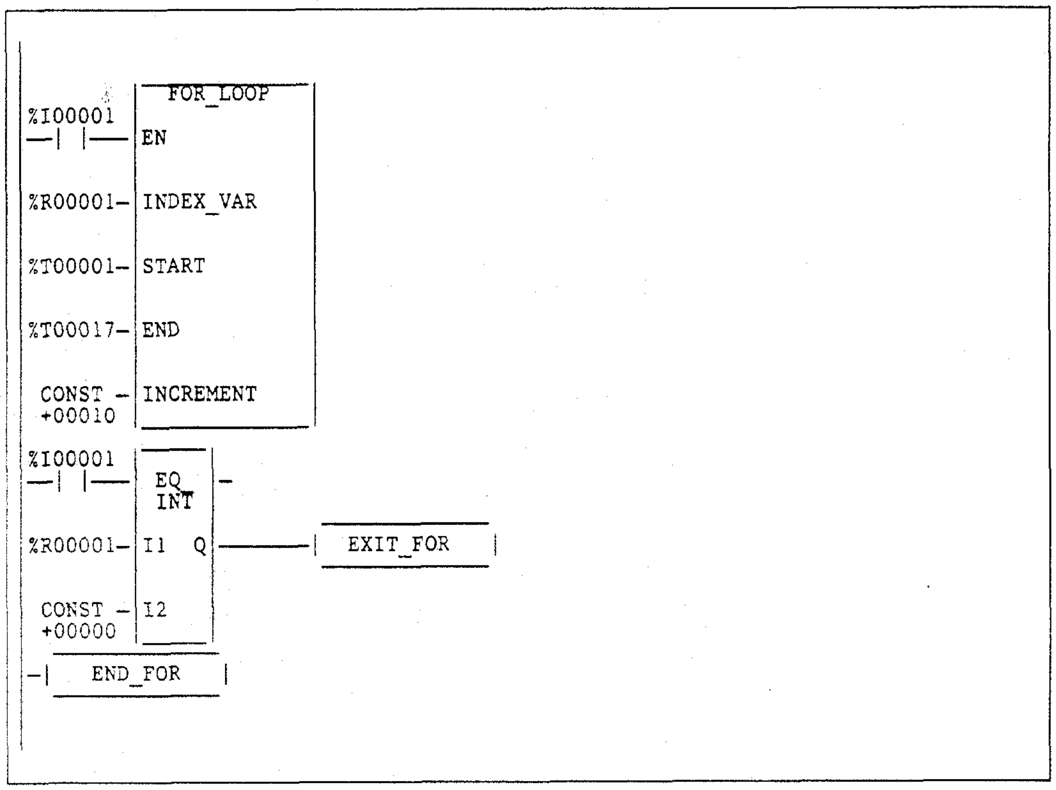
Example 1:

In the first example, the value for %M00001 (START) is 1, and the value for %M00017 (END) is 10. The INDEX_VAR (%R00001) will increment by the INCREMENT (assumed to be 1 when omitted) starting at 1 until it reaches the ending value 10. The ADD function of the loop is executed 10 times, adding the current value of I1 (%R00001) 1 ... 10 to I2 (%R00002).



Example 2:

In this next example, the value for %T00001 (START) is —100, and the value for %T00017 (END) is 100. The INDEX_VAR (%R00001) will increment by tens, starting at —100 until it reaches it end value of +100. The EQ function of the loop will try to execute 21 times, with I1 (%R00001) equal to −100, −90, −80, −70, −60, −50, −40, −30, −20, −10, 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100. When I1 (%R00001) is 0, the EXIT statement will be enabled and power flow will jump directly to the statement after the END_FOR statement.



SVCREQ

Use the Service Request (SVCREQ) function to request one of the following special PLC services:

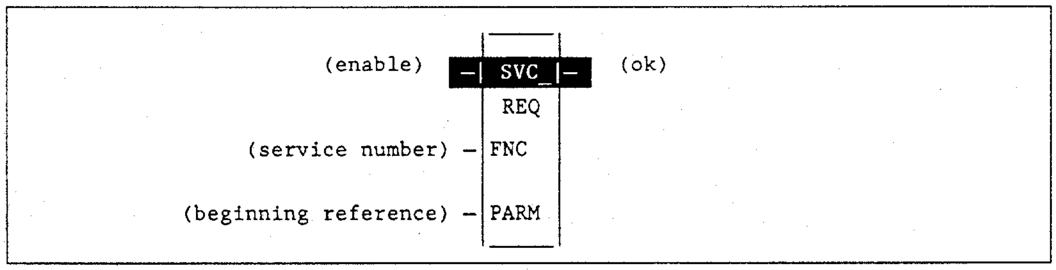
Table 4-3. Service Request Functions

Function	Description	Page
1	Change/Read Constant Sweep Timer.	4-191
2	Read Window Values.	4-194
3	Change Programmer Communications Window State and Values.	4-195
4	Change System Communications Window State and Values.	4-196
5	Reserved.	–
6	Change/Read Checksum Task State and Number of Words to Checksum.	4-200
7	Change/Read Time-of-Day Clock State and Values.	4-202
8	Reset Watchdog Timer.	4-206
9	Read Sweep Time from Beginning of Sweep.	4-207
10	Read Folder Name.	4-208
11	Read PLC ID.	4-209
12	Read PLC Run State.	4-210
13	Shut Down PLC.	4-211
14	Clear Fault Tables.	4-212
15	Read Last-Logged Fault Table Entry.	4-213
16	Read Elapsed Time Clock.	4-217
17	Mask/Unmask I/O Interrupt.	4-218
18	Read I/O Override Status.	4-220
19	Set Run Enable/Disable.	4-221
20	Read Fault Tables.	4-222
21	Log User-Defined PLC Fault.	4-226
22	Mask/Unmask Timed Interrupts.	4-228
23	Read Master Checksum.	4-229
24	Reserved.	–
25	Disable/Enable EXE Block and Standalone C Program Checksums	4-231
26	Switch Roles (Active to Backup and Backup to Active).	4-232
27	Write to Reverse Transfer Area.	4-233
28	Read from Reverse Transfer Area.	4-233
29–31	Reserved.	–
32	Suspend/Resume I/O Interrupt.	4-235

The SVCREQ function has three input parameters and one output parameter. When the SVCREQ receives power flow, the PLC is requested to perform the function (FNC) indicated. Parameters for the function begin at the reference given for PARM. The SVCREQ function passes power flow unless an incorrect function number, incorrect parameters, or out-of-range references are specified. Additional causes for failure are described on the pages that follow.

The reference given for PARM is the first of a group that make up the "parameter block" for the function. Successive 16-bit locations store additional parameters. The total number of references required will depend on the type of SVCREQ function being used.

Parameter blocks may be used as both inputs for the function and the location where data may be output after the function executes. Therefore, data returned by the function is accessed at the same location specified for PARM.



Parameters:

Parameter	Description
enable	When enable is energized, the requested service is performed.
FNC	FNC contains the constant or reference for the requested service.
PARM	PARM contains the beginning reference for the parameter block for the requested service.
ok	The ok output is energized when the function is performed without error.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
FNC		•	•	•	•		•		•	•	•	•	•	•	•	
PARM		•	•	•	•		•		•	•	•	•	•	•		
ok	•															•

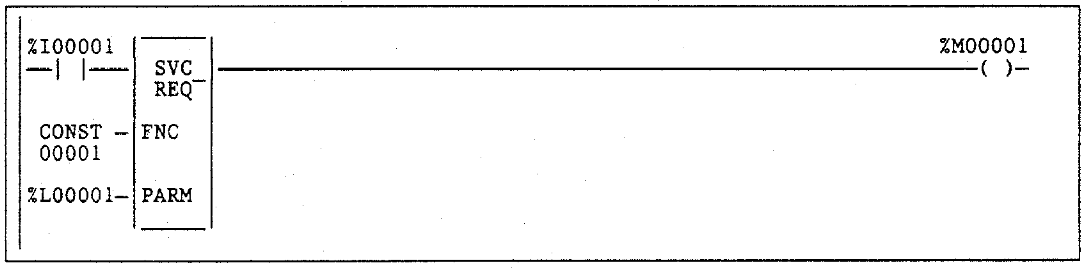
Note: Indirect referencing is available for all register references (%R, %P, %L, %AI, %AQ, and %UR).
• Valid reference or place where power may flow through the function.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

Example:

In the following example, when the enabling input %I00001 is ON, SVCREQ function number 1 is called, with the parameter block located starting at %L00001. Output coil %M00001 is set ON if the operation succeeds.



SVCREQ #1: Change/Read Constant Sweep Timer

Use SVCREQ function #1 to:

- Disable **CONSTANT SWEEP** mode.
- Enable **CONSTANT SWEEP** mode and use the old timer value.
- Enable **CONSTANT SWEEP** mode and use a new timer value.
- Set a new timer value only.
- Read **CONSTANT SWEEP** mode state and timer value.

The parameter block has a length of two words.

To disable **CONSTANT SWEEP** mode, enter SVCREQ function #1 with this parameter block:

0	address
ignored	address + 1

To enable **CONSTANT SWEEP** mode, enter SVCREQ function #1 with this parameter block:

1	address
0 or timer value	address + 1

Note

If the timer should use a new value, enter it in the second word. If the timer value should not be changed, enter 0 in the second word. If the timer value does not already exist, entering 0 will cause the function to set the OK output to OFF.

To change the timer value <u>**without**</u> changing the selection for sweep mode state, enter SVCREQ function #1 with this parameter block:

2	address
new timer value	address + 1

To read the current timer state and value without changing either, enter SVCREQ function #1 with this parameter block:

3	address
ignored	address + 1

Note

After using SVCREQ function #1 with the parameter block on the previous page, Release 6 and higher CPUs will provide the **return** values 0 for Normal Sweep, 1 for Constant Sweep, or 2 for Microcycle. Do not confuse this with the *input* values shown below.

Successful execution will occur, unless:

1. A number other than 0, 1, 2, or 3 is entered as the requested operation:

0	Disable **CONSTANT SWEEP** mode.
1	Enable **CONSTANT SWEEP** mode.
2	Set a new timer value only.
3	Read **CONSTANT SWEEP** mode and timer value. (See Note above).

2. The time value is greater than 2550 ms (2.55 seconds).
3. Constant sweep time is enabled with no timer value programmed, or with an old value of 0 for the timer.

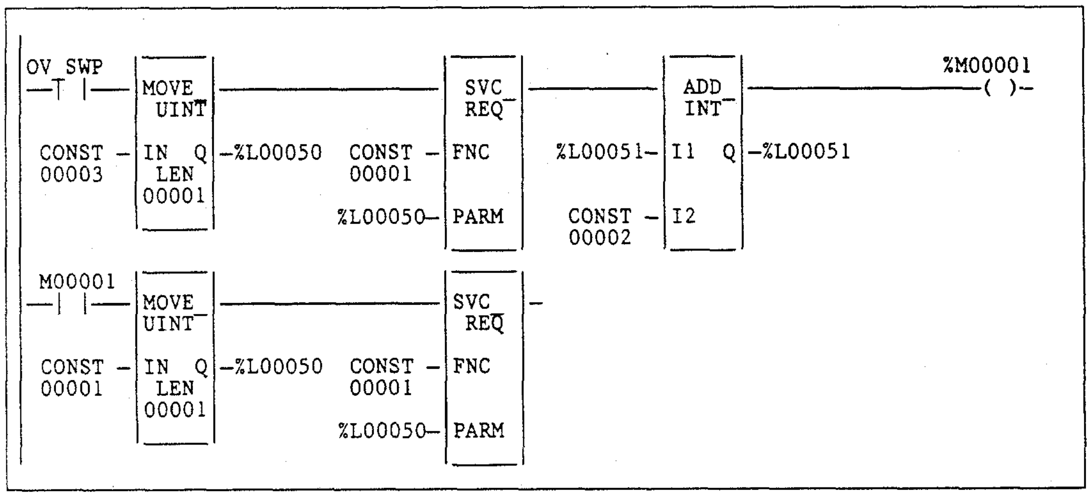
After the function executes, the function returns the timer state and value in the same parameter block references:

0 = disabled	
1 = enabled	address
current timer value	address + 1

If word address + 1 contains the hexadecimal value FFFF, no timer value has ever been programmed.

Example:

This example shows logic in a program block. When enabling contact OV_SWP is set, the constant sweep timer is read, the timer is increased by two milliseconds, and the new timer value is sent back to the PLC. The parameter block is in local memory at location %L00050. Because the MOVE and ADD functions require three horizontal contact positions, the example logic uses discrete internal coil %M00001 as a temporary location to hold the successful result of the first rung line. On any sweep in which OV_SWP is not set, %M00001 is turned off.



SVCREQ #2: Read Window Values

Use SVCREQ function #2 to obtain the current window mode time values for the programmer communications window, the system communications window, and the background task window. There are three modes for each window:

Mode Name	Value	Description
Limited Mode	0	The execution time of the window is limited to its respective default value or to a value defined using SVCREQ function #3 for the programmer communications window or SVCREQ function #4 for the systems communications window. The window will terminate when it has no more tasks to complete.
Constant Mode	1	Each window will operate in a **RUN TO COMPLETION** mode, and the PLC will alternate among the three windows for a time equal to the sum of each window's respective time value. If one window is placed in **CONSTANT** mode, the remaining two windows are automatically placed in **CONSTANT** mode. If the PLC is operating in **CONSTANT WINDOW** mode and a particular window's execution time is not defined using the associated SVCREQ function, the default time for that window is used in the constant window time calculation.
Run to Completion Mode	2	Regardless of the window time associated with a particular window, whether default or defined using a service request function, the window will run until all tasks within that window are completed.

A window is disabled when the time value is zero.

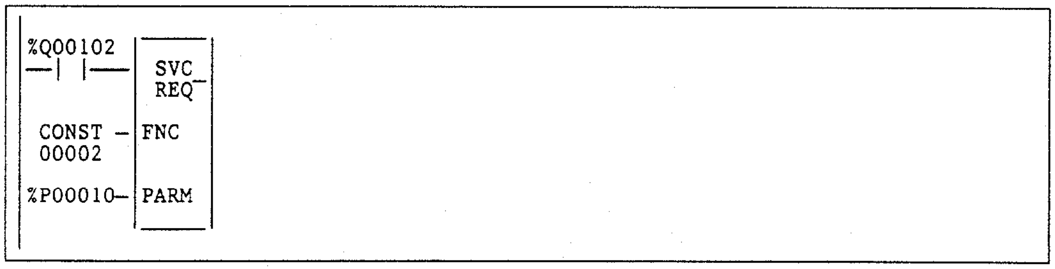
The parameter block has a length of three words:

	High Byte	Low Byte	
Programmer Window	Mode	Value in ms	address
System Communications Window	Mode	Value in ms	address + 1
Background Window	Mode	Value in ms	address + 2

All parameters are output parameters. It is not necessary to enter values in the parameter block to program this function. Output values for all three windows are given in milliseconds.

Example:

In the following example, when enabling output %Q00102 is set, the PLC operating system places the current time values of the three windows in the parameter block starting at location %P00010. Additional examples showing the Read Window Values function are included in the next three SYS REQ function descriptions.



SVCREQ #3: Change Programmer Communications Window Mode and Timer Value

Use SVCREQ function #3 to change the programmer communications window mode and timer value. The change will occur in the CPU sweep following the sweep in which the function is called.

The SVCREQ function #3 will pass power flow to the right unless one of the following occurs:

1. A mode other than 0 (Limited), 1 (Constant), or 2 (Run-to-Completion) is selected.
2. The PLC is in Microcycle Sweep Mode and a mode other than 1 (Constant) is selected.

The parameter block has a length of one word.

To disable the programmer window, enter SVCREQ function #3 with this parameter block:

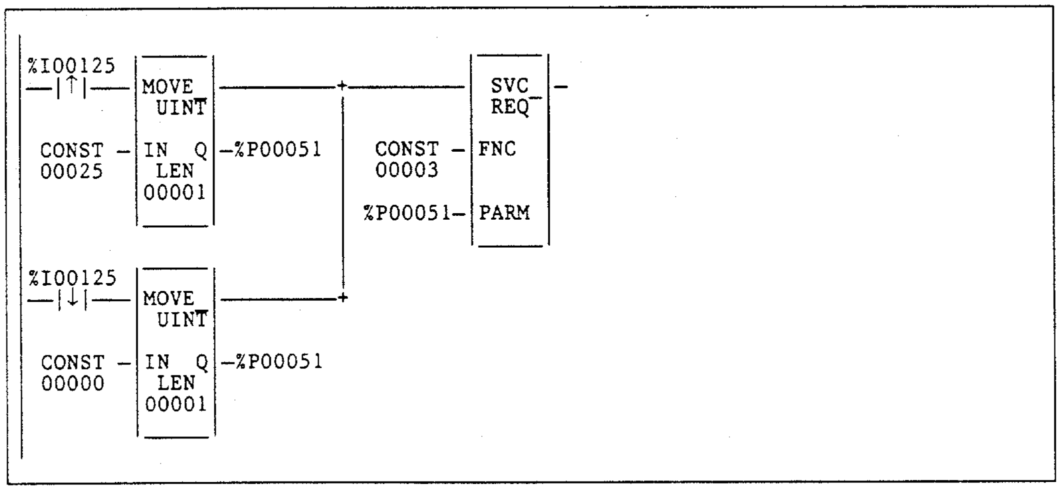
High Byte	Low Byte	
0	0	address

To enable the programmer window, enter SVCREQ function #3 with this parameter block:

High Byte	Low Byte	
Mode	Value from 1 to 255 ms	address

Example:

In the following example, when enabling input %I00125 transitions on the programmer communications window is enabled in Limited mode with a timer value of 25 ms. When the contact transitions off, the window is disabled. The parameter block is in global memory location %P00051.



SVCREQ #4: Change System Communications Window Mode and Timer Value

Use SVCREQ function #4 to change the system communications window mode and timer value. The change will occur in the CPU sweep following the sweep in which the function is called.

The SVCREQ function #4 will pass power flow to the right unless one of the following occurs:

1. A mode other than 0 (Limited), 1 (Constant), or 2 (Run-to-Completion) is selected.
2. The PLC is in Microcycle Sweep Mode and a mode other than 1 (Constant) is selected.

The parameter block has a length of one word.

To disable the system communications window, enter SVCREQ function #4 with this parameter block:

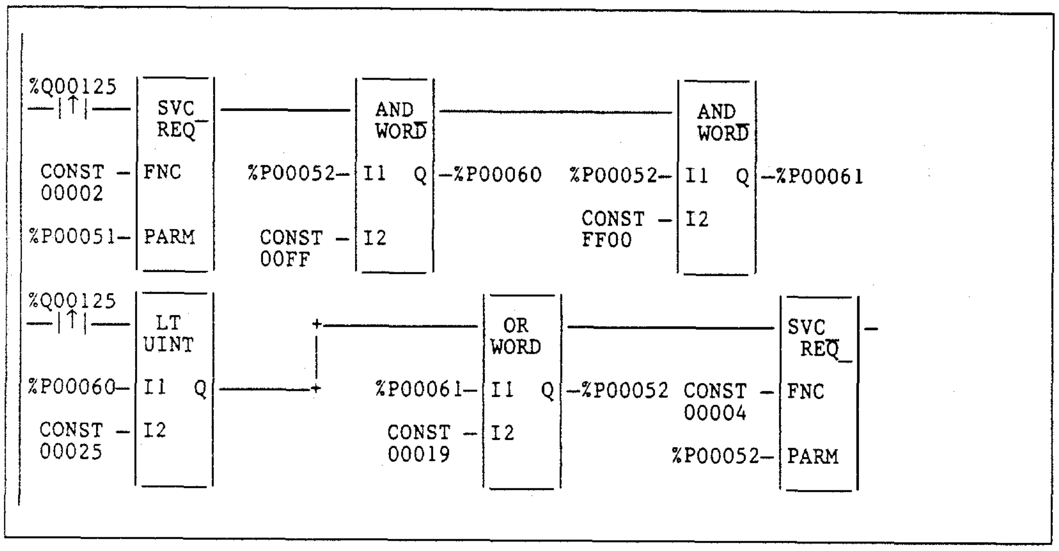
High Byte	Low Byte	
0	0	address

To enable the system communications window, enter SVCREQ function #4 with this parameter block:

High Byte	Low Byte	
Mode	Value from 1 to 255 ms	address

Example:

In the following example, when enabling output %Q00125 transitions on the mode and timer value of the system communications window is read. If the timer value is greater than or equal to 25 ms, the value is not changed. If it is less than 25 ms, the value is changed to 25 ms. In either case, when the rung completes execution the window is enabled. The parameter block for all three windows is at location %P00051. Since the mode and timer for the system communications window is the second value in the parameter block returned from the Read Window Values function (function #2), the location of the existing window time for the system communications window is in the low byte of %P00052.


%Q00125
SVC REQ
CONST 00002
FNC
%P00051
PARM
AND WORD
%P00052
I1
Q
%P00060
CONST 00FF
I2
AND WORD
%P00052
I1
Q
%P00061
CONST FF00
I2
%Q00125
LT UINT
%P00060
I1
Q
CONST 00025
I2
OR WORD
%P00061
I1
Q
%P00052
CONST 00019
I2
SVC REQ
CONST 00004
FNC
%P00052
PARM

SVCREQ #5: Change Background Task Window Mode and Timer Value

Use the SVCREQ function # 5 to change the background task window mode and timer value. The change will occur during the same CPU sweep in which the function is called.

Use SVCREQ function #4 to enable or disable the system communications window. The change will occur in the same CPU sweep in which the function is called.

The SVCREQ function #4 will pass power flow to the right unless one of the following occurs:

1. A mode other than 0 (Limited), 1 (Constant), or 2 (Run-to-Completion) is selected.
2. The PLC is in Microcycle Sweep Mode and a mode other than 1 (Constant) is selected.

SVCREQ function #5 always passes power flow to the right. The parameter block has a length of 1 word.

To disable the background task window, enter SVCREQ function 5 with this parameter block:

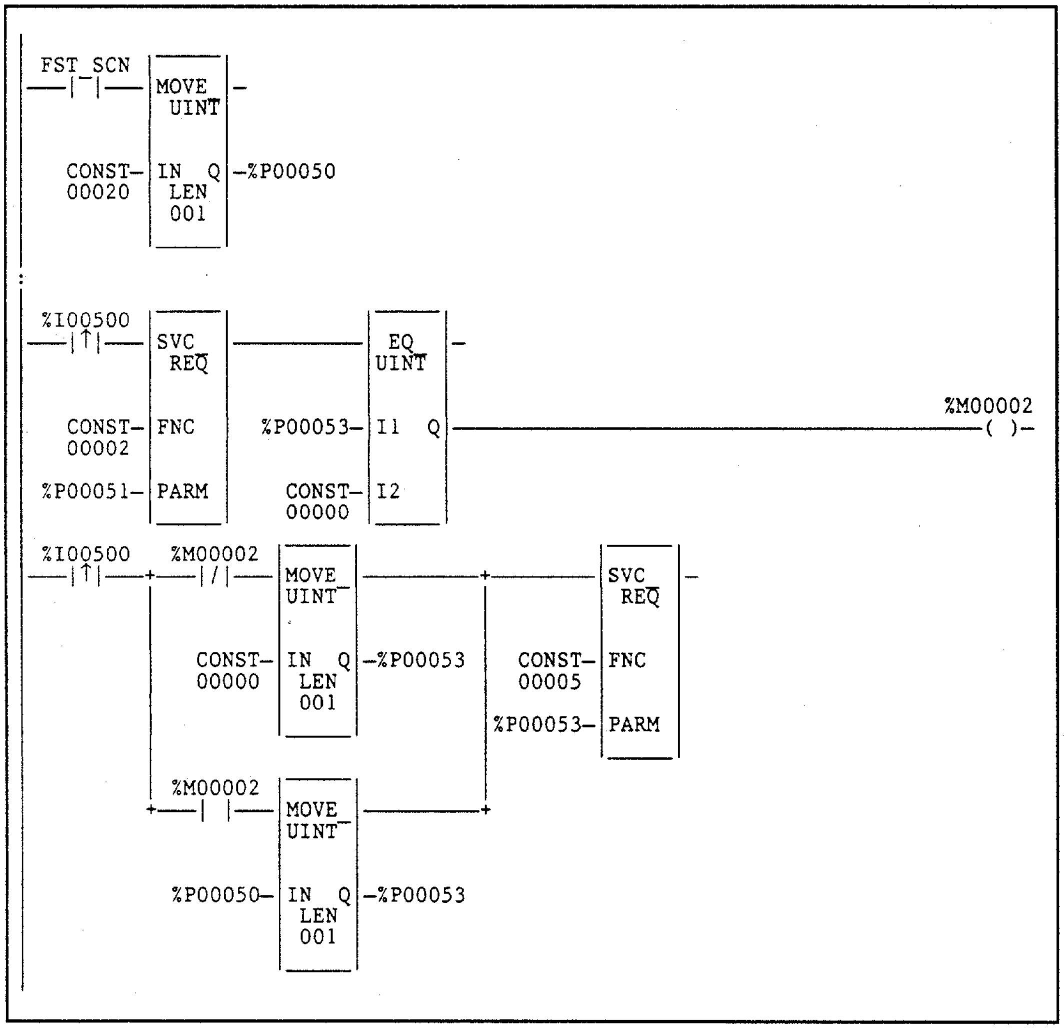
High Byte	Low Byte	
0	0	address

To enable the background task window, enter SVCREQ function #5 with this parameter block:

High Byte	Low Byte	
Mode	Value from 1 to 255 ms	address

Example:

In the following example, when enabling contact FST_SCN is set in the first scan, the MOVE function establishes a default mode of Limited and a default timer value of 20 ms for the background task window, using a parameter block beginning at %P00050. Later in the program when input %I00500 transitions on, the state of the background task window toggles on and off. The parameter block for all three windows is at location %P00051. The mode and timer value for the background task window is the third value in the parameter block returned from the Read Window Values function (function #2); therefore, the location of the existing window time for the system communications window is in the low byte of %P00053.



SVCREQ #6: Change/Read Checksum Task State and Number of Words to Checksum

Use SVCREQ function #6 to read the current word count or set a new word count. By default, 16 words will be checked. Successful execution will occur, unless some number other than 0 or 1 is entered as the requested operation (see below).

The parameter block has a length of two words.

To read the current word count, enter SVCREQ function #6 with this parameter block:

0	address
ignored	address + 1

After the function executes, the function returns the current checksum in the second word of the parameter block. No range is specified for the read function; the value returned is the number of words currently being checksummed.

0	address
current word count	address + 1

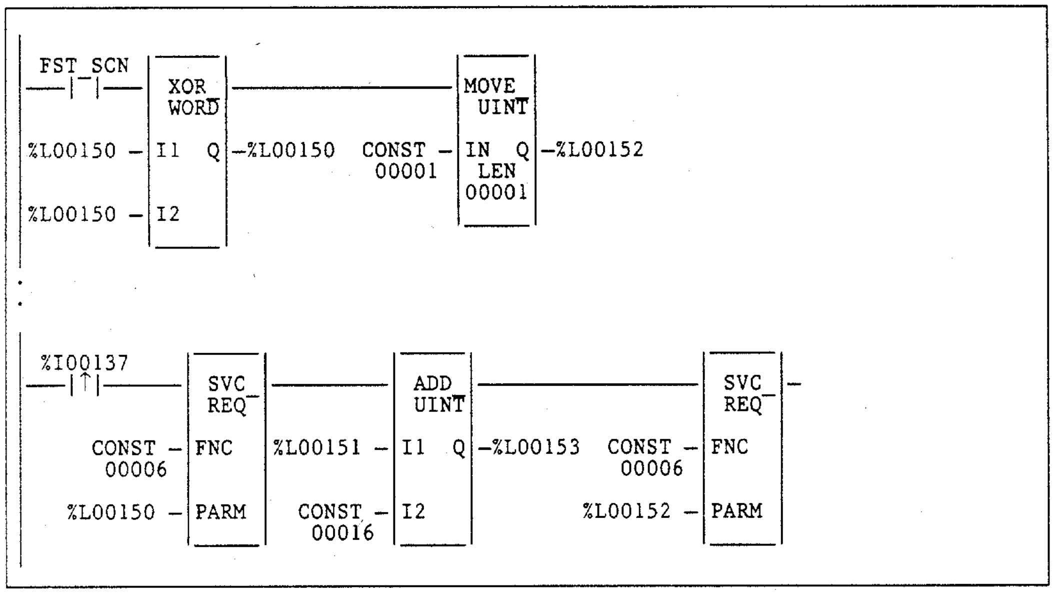
To set a new word count, enter SVCREQ function #6 with this parameter block:

1	address
new word count	address + 1

Entering 1 causes the PLC to adjust the number of words to be checksummed to the value given in the second word of the parameter block, rounded up to a multiple of 8. To disable checksumming, set the count to zero.

Example:

In the following example, when enabling contact FST_SCN is set, the parameter blocks for the checksum task function are built. Later in the program when input %I00137 transitions on, the number of words being checksummed is read from the PLC operating system. This number is increased by 16, with the results of the ADD_UINT function being placed in the "hold new count for set" parameter. The second service request block requests the PLC to set the new word count.



The example parameter blocks are located at address %L00150. They have the following content:

0 = read current count	%L00150
hold current count	%L00151

1 = set current count	%L00152
hold new count for set	%L00153

SVCREQ #7: Change/Read Time-of-Day Clock State and Values

Use SVCREQ function #7 to read or set the time-of-day clock in the PLC.

The length of the parameter block depends on the data format. Numeric and unpacked BCD each require nine words. BCD format requires six words. Packed ASCII requires twelve words.

0 = read time and date 1 = set time and date	address (word 1)
0 = numeric data format 1 = BCD format 2 = unpacked BCD format 3 = packed ASCII format	address + 1 (word 2)
data	address + 2 to end (word 3)

In word 1, specify whether the function should read or change the values.

0	=	read
1	=	change (write)

In word 2, specify a data format (i.e., the format used to read or write):

0	=	numeric
1	=	BCD
2	=	unpacked BCD
3	=	packed ASCII with embedded spaces and colons

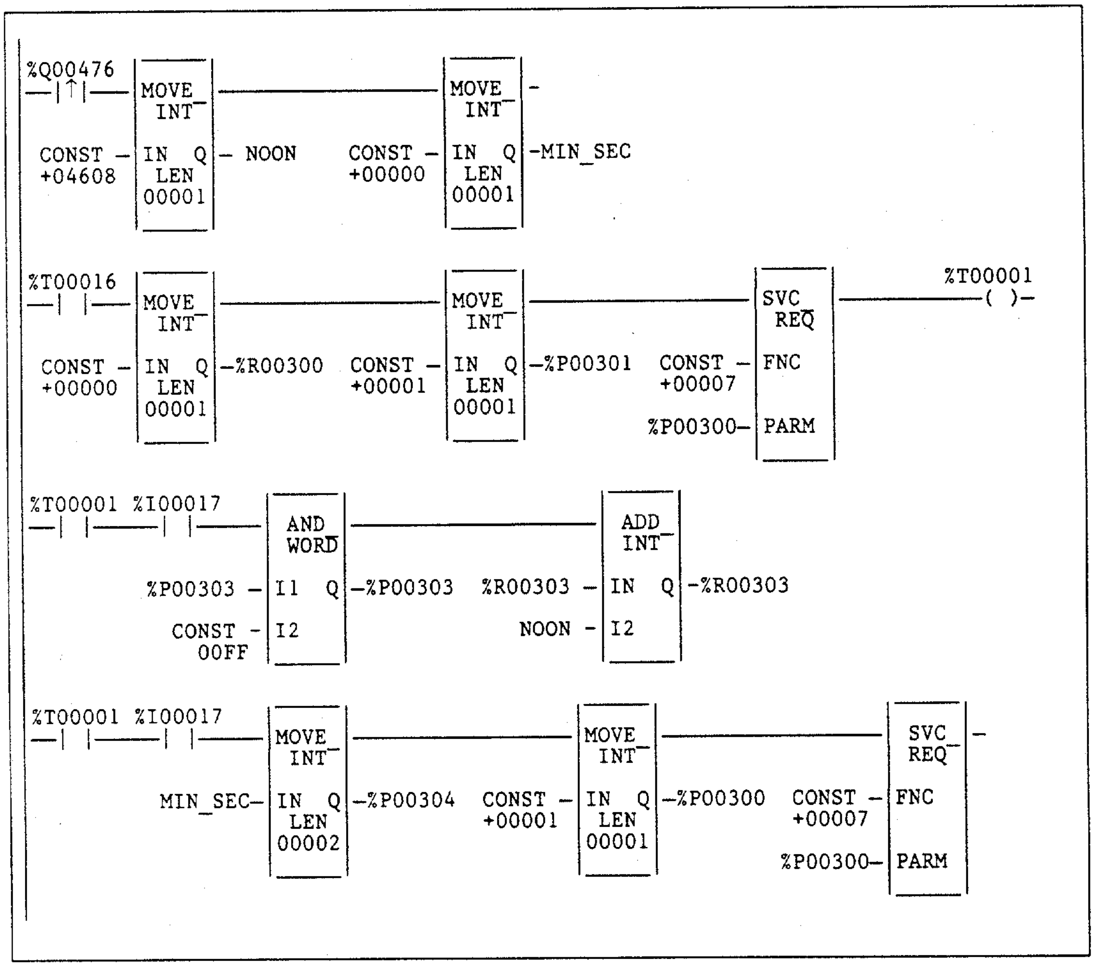
Words 3 to the end of the parameter block contain output data returned by a read function, or new data being supplied by a change function. In both cases, format of these data words is the same. When reading the date and time, words (address + 2) through (address + 8) of the parameter block are ignored on input.

Successful execution will occur unless:

- Some number other than 0 or 1 is entered as the requested operation (see below).
- An invalid data format is specified.
- The data provided is not in the expected format.

Example:

In the following example, when output %Q00476 is on, a parameter block for the time-of-day clock is built to first request the current date and time, and then set the clock to 12 noon using the BCD format. The parameter block is located at global data location %P00300. Array NOON has been set up elsewhere in the program to contain the values 12, 0, and 0. (Array NOON must also contain the data at %R0300.) The BCD format requires six contiguous memory locations for the parameter block.



Parameter Block Contents

Parameter block contents for the four different data formats are shown on the following pages. For all data formats, hours are stored in a 24-hour format, and the day of the week is a numeric value.

Value	Day of the Week
1	Sunday
2	Monday
3	Tuesday
4	Wednesday
5	Thursday
6	Friday
7	Saturday

To Change/Read Date and Time Using Numeric Format:

In numeric format, year, month, day of month, hours, minutes, seconds and day of week each occupy one unsigned integer. To read and/or change the date and time using numeric format, enter SVCREQ function #7 with this parameter block:

High Byte	Low Byte		Example output parameter block: Read Date and Time in Numeric Format (Weds, June 15, 1988 at 12:15:30)
1 = change or	0 = read	address	0
0		address + 1	0
	year	address + 2	88
	month	address + 3	06
	day of month	address + 4	15
	hours	address + 5	12
	minutes	address + 6	15
	seconds	address + 7	30
	day of week	address + 8	04

To Change/Read Date and Time Using BCD Format:

In BCD format, each of the time and date items occupies a single byte. This format requires six words. The last byte of the sixth word is not used. When setting the date and time, this byte is ignored; when reading date and time, the function returns a null character (00).

Example output parameter block:
Read Date and Time in BCD Format
(Mon, July 3, 1988 at 2:45:30 p.m.)

High Byte	Low Byte		Example (high)	Example (low)
1 = change or	0 = read	address		0
1		address + 1		1
month	year	address + 2	07	88
hours	day of mo.	address + 3	14	03
seconds	minutes	address + 4	30	45
(null)	day of week	address + 5	00	02

To Change/Read Date and Time Using Unpacked BCD Format:

In Unpacked BCD format, each digit of the time and date items occupies the low order four bits of a byte. The upper four bits of each byte are always zero. This format requires nine words.

High Byte	Low Byte	
1 = change or	0 = read	address
2		address + 1
	year	address + 2
	month	address + 3
	day of month	address + 4
	hours	address + 5
	minutes	address + 6
	seconds	address + 7
	day of week	address + 8

Example output parameter block:
Read Date and Time in Unpacked BCD Format
(Thurs, Dec. 28, 1989 at 9:34:57)

0h	
2h	
08h	09h
01h	02h
02h	08h
00h	09h
03h	04h
05h	07h
00h	05h

To Change/Read Date and Time Using Packed ASCII with Embedded Colons Format:

In Packed ASCII format, each digit of the time and date items is an ASCII formatted byte. In addition, spaces and colons are embedded into the data to permit it to be transferred unchanged to a printing or display device. This format requires 12 words.

High Byte	Low Byte	
1 = change or	0 = read	address
3		address + 1
year	year	address + 2
month	(space)	address + 3
(space)	month	address + 4
day of mo.	day of mo.	address + 5
hours	(space)	address + 6
:	hours	address + 7
minutes	minutes	address + 8
seconds	:	address + 9
(space)	seconds	address + 10
day of week	day of week	address + 11

Example output parameter block:
Read Date and Time in Packed ASCII Format
(Tues, Oct. 2, 1989 at 23:13:00)

0h	
3h	
39h	38h
31h	20h
20h	30h
32h	30h
32h	20h
3Ah	33h
33h	31h
30h	3Ah
20h	30h
33h	30h

SVCREQ #8: Reset Watchdog Timer

Use SVCREQ function #8 to reset the watchdog timer during the sweep. When the watchdog timer expires, the PLC shuts down without warning. This function allows the timer to keep going during a time-consuming task (for example, while waiting for a response from a communications line).

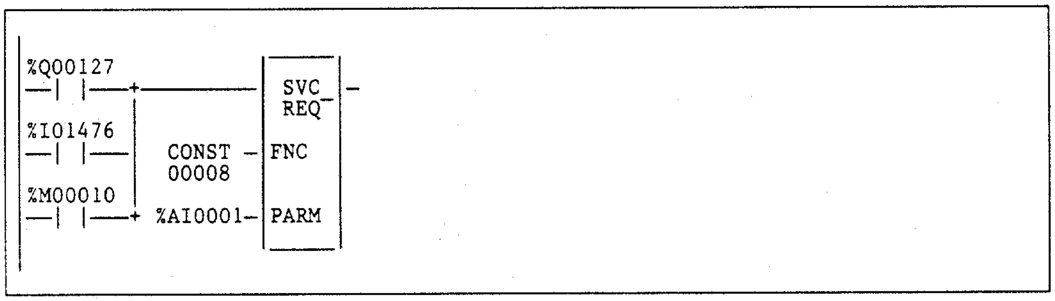
Caution

Be sure that restarting the watchdog timer does not adversely affect the controlled process.

This function has no associated parameter block; however, the programming software requires that an entry be made for PARM. Enter any appropriate reference here; it will not be used.

Example:

In the following example, when enabling output %Q00127 or input %I01476 or internal coil %M00010 is set, the watchdog timer is reset.



SVCREQ #9: Read Sweep Time from Beginning of Sweep

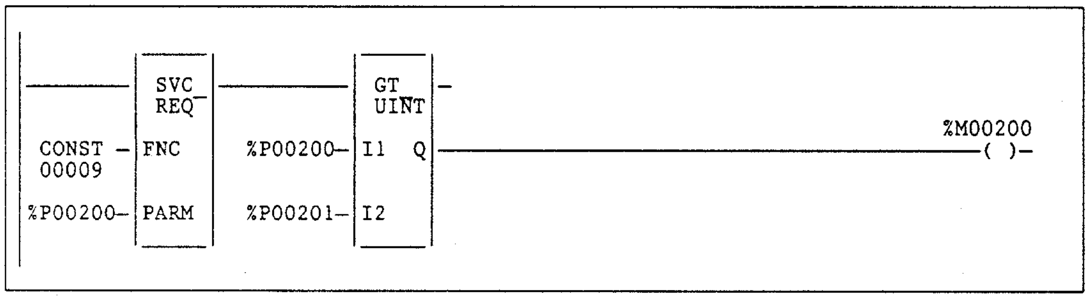
Use SVCREQ function #9 to read the time in milliseconds since the start of the sweep. The data format is in unsigned 16-bit integer.

The parameter block is an output parameter block only; it has a length of one word.

time since start of sweep	address

Example:

In the following example, the elapsed time from the start of the sweep is always read into location %P00200. If it is greater than the value in %P00201, internal coil %M00200 is turned on.



Note

This SVCREQ has a different meaning with Microcycle mode. It reflects the time from the beginning of the sweep in which the program was scheduled to begin execution.

SVCREQ #10: Read Folder Name

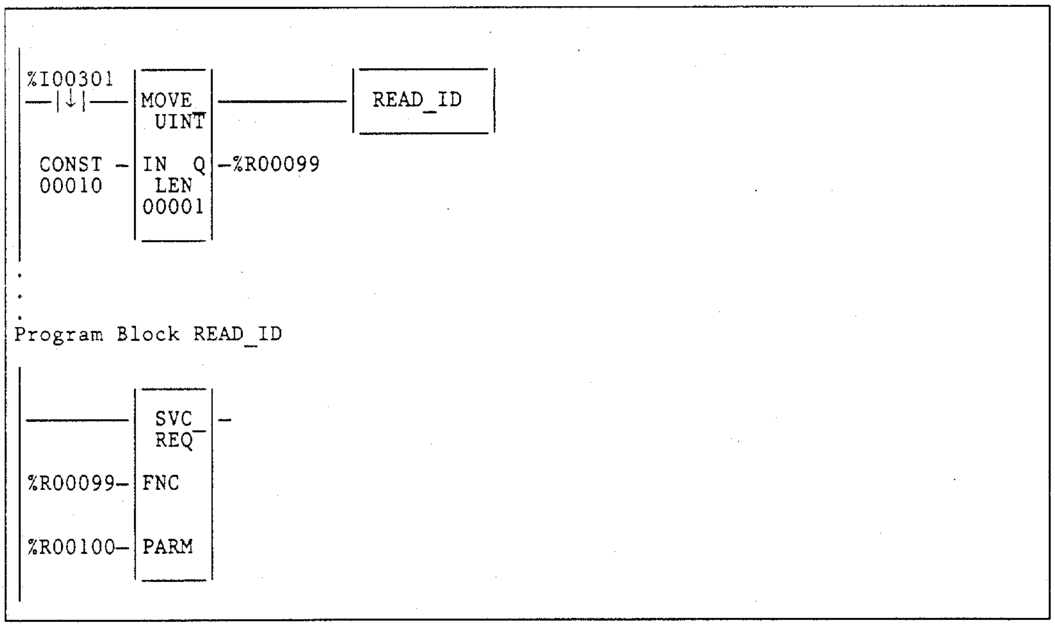
Use SVCREQ function #10 to read the name of the currently-executing folder.

The output parameter block has a length of four words. It returns eight ASCII characters; the last is a null character (00h). If the program name has fewer than seven characters, null characters are appended to the end.

Low Byte	High Byte	
character 1	character 2	address
character 3	character 4	address + 1
character 5	character 6	address + 2
character 7	00	address + 3

Example:

In the following example, when enabling input %I00301 transitions off, register location %R00099 is loaded with the value 10, which is the function code for the Read Folder Name function. The Program Block READ_ID is then called to actually retrieve the folder name. The parameter block is located at address %R00100. READ_ID is also used in the next example.



SVCREQ #11: Read PLC ID

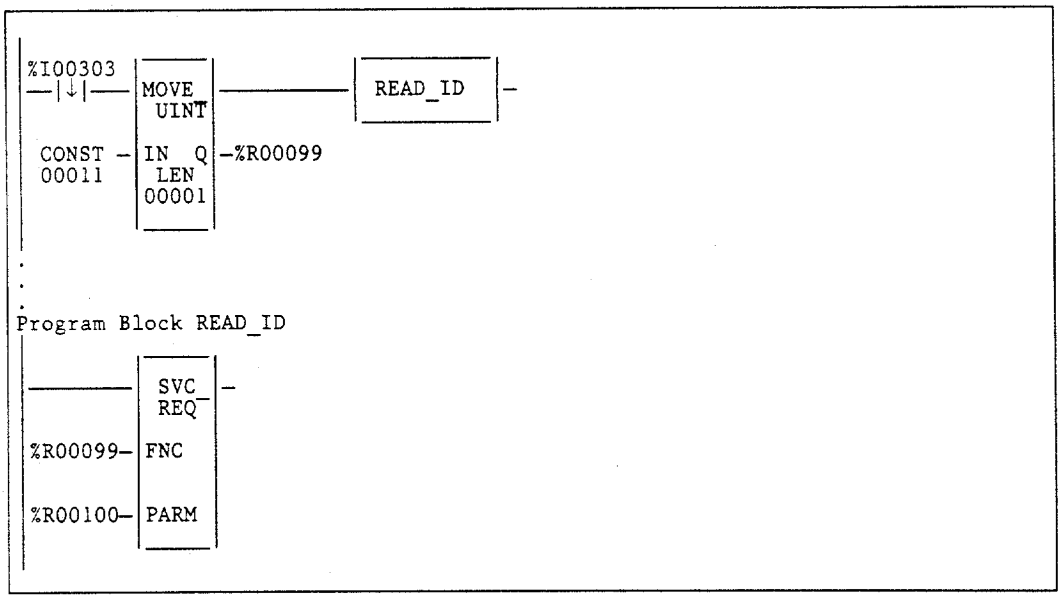
Use SVCREQ function #11 to read the name of the Series 90 PLC executing the program.

The output parameter block has a length of four words. It returns eight ASCII characters; the last is a null character (00h). If the PLC ID has fewer than seven characters, null characters are appended to the end.

Low Byte	High Byte	
character 1	character 2	address
character 3	character 4	address + 1
character 5	character 6	address + 2
character 7	00	address + 3

Example:

In the following example, when enabling input %I00302 transitions off, register location %R00099 is loaded with the value 11, which is the function code for the Read PLC ID function. The program block READ_ID is then called to actually retrieve the ID. The parameter block is located at address %R00100. Except for the enabling contact and function number, this is the same code used in the previous example.



SVCREQ #12: Read PLC Run State

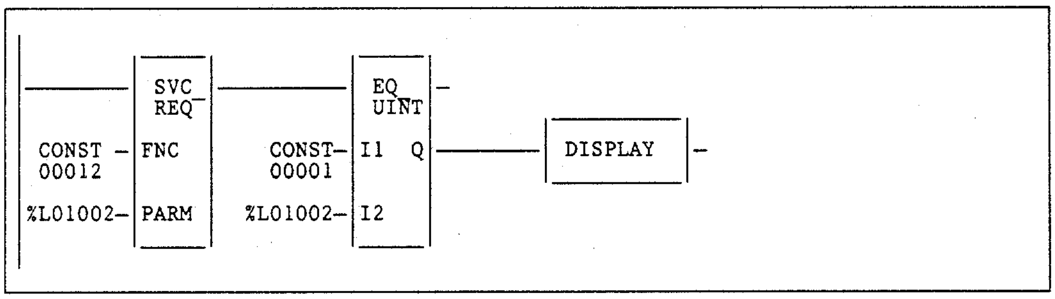
Use SVCREQ function #12 to read the current RUN state of the PLC CPU.

The parameter block is an output parameter block only; it has a length of one word.

1 = run/disabled	address
2 = run/enabled	

Example:

In the following example, the PLC run state is always read into location %L01002. If the state is Run/Disabled, the CALL function calls program block DISPLAY.



SVCREQ #13: Shut Down (Stop) PLC

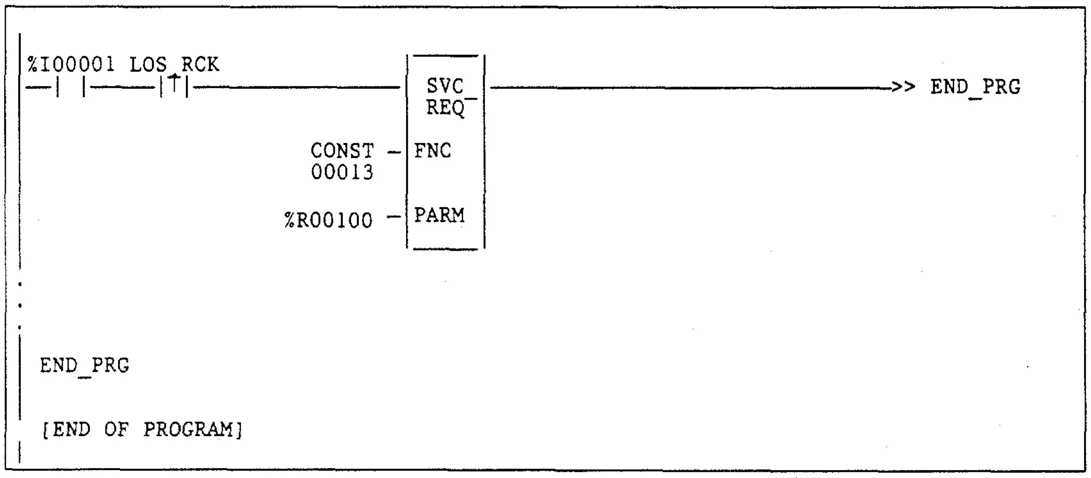
Use SVCREQ function #13 to stop the PLC at the end of the current sweep. All outputs will go to their designated default states at the beginning of the next PLC sweep. An informational fault is placed in the PLC fault table, noting that a "SHUT DOWN PLC" function block was executed.

This function has no parameter block; however, Logicmaster 90-70 software requires an entry be made for PARM.

Example:

In the following example, when enabling input %I00001 is set and a lost rack fault occurs, SVCREQ function #13 executes. Since no parameter block is needed, the PARM input is not used; however, the programming software requires that an entry be made for PARM.

This example uses a JUMP to the end of the program to force a shutdown if the Shutdown PLC function executes successfully. This JUMP and LABEL are needed because the transition to **STOP** mode does not occur until the end of the sweep in which the function executes.



SVCREQ #14: Clear Fault Tables

Use SVCREQ function #14 to clear either the PLC or I/O fault table. The SVCREQ output is set ON, unless some number other than 0 or 1 is entered as the requested operation (see below).

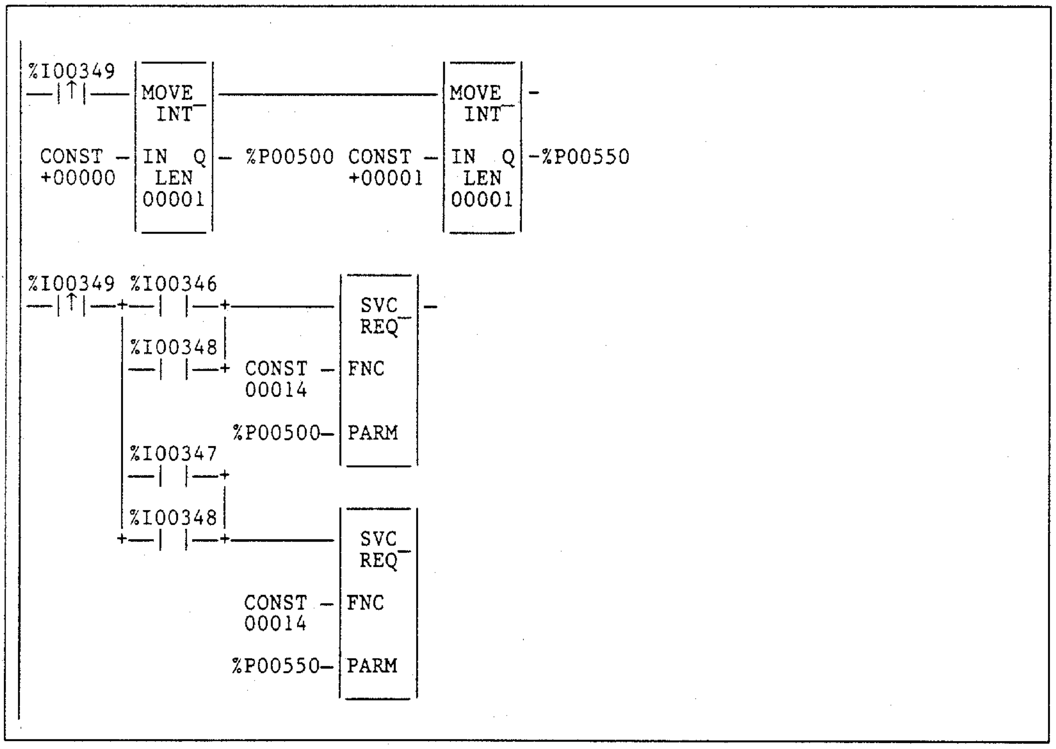
The parameter block is an input parameter block only; it has a length of one word.

0 = Clear PLC fault table.	address
1 = Clear I/O fault table.	

Example:

In the following example, when input %I00346 is on and input %I00349 transitions on, the PLC fault table is cleared. When input %I00347 is on and input %I00349 transitions on, the I/O fault table is cleared. When input %I00348 is on and input %I00349 transitions on, both are cleared.

The parameter block for the PLC fault table is located at %P00500; for the I/O fault table, the parameter block is located at %P00550. Both parameter blocks are set up elsewhere in the program.



SVCREQ #15: Read Last-Logged Fault Table Entry

Use SVCREQ function #15 to read the last entry logged in either the PLC or I/O fault table. The SVCREQ output is set ON, unless some number other than 0, 1, 80, or 81 (hexadecimal base) is entered as the requested operation (see below) or the fault table is empty. (For more information on fault table entries, refer to chapter 3, "Fault Explanation and Correction.")

The parameter block has a length of 22 words. The input parameter block has this format:

0 = Read PLC fault table.
1 = Read I/O fault table.
80h = Read extended PLC fault table.
81h = Read extended I/O fault table.

The format for the output parameter block depends on whether the function reads data from the PLC fault table, the I/O fault table, the extended PLC fault table (see next page), or the extended I/O fault table (see next page).

PLC Fault Table Output Format			I/O Fault Table Output Format	
High Byte	Low Byte		High Byte	Low Byte
0		address	1	
	long/short	address + 1		long/short
spare		address + 2		reference
PLC fault address		address + 3		
		address + 4	I/O fault address	
fault group and action		address + 5		
error code		address + 6	fault group and action	
fault specific data		address + 7	fault type	fault category
		address + 8	fault specific data	fault description
		address + 9		
		address + 10		
		address + 11		
		address + 12		
		address + 13		
		address + 14		
		address + 15		
		address + 16		
		address + 17		
		address + 18		
time stamp		address + 19	time stamp	
		address + 20		
		address + 21		

In the least significant byte of word address + 1, the Long/Short indicator defines the quantity of fault specific data present in the fault entry.

As mentioned on the previous page, the format for the output parameter block depends on whether the function reads data from the PLC fault table, the I/O fault table, the extended PLC fault table, or the extended I/O fault table. The extended versions are shown below.

Extended PLC Fault Table Output Format | **Extended I/O Fault Table Output Format**

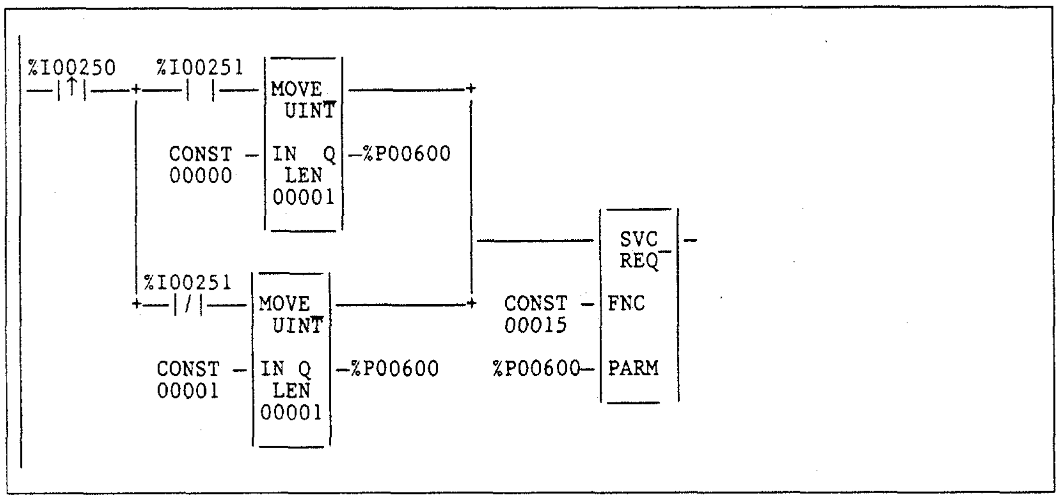
High Byte	Low Byte		High Byte	Low Byte
80h		address	81h	
	long/short	address + 1		long/short
spare		address + 2	reference	
PLC fault address		address + 3		
		address + 4	I/O fault address	
fault group and action		address + 5		
error code		address + 6	fault group and action	
fault specific data		address + 7	fault type	fault category
		address + 8		fault description
		address + 9	fault specific data	
		address + 10		
		address + 11		
		address + 12		
		address + 13		
		address + 14		
		address + 15		
		address + 16		
		address + 17		
		address + 18		
time stamp		address + 19	time stamp	
		address + 20		
		address + 21		
		address + 22		
reserved		address + 23	reserved	

Since the extended fault format is 2 words (4 bytes) longer than the standard format, the amount of reference memory needed to read the last fault has increased. The following table shows the number of bytes needed to read the last fault entry in each of the fault formats.

Fault Table	Number of Bytes
PLC fault table	00 = 44 bytes
Extended PLC fault table	80 = 48 bytes
I/O fault table	01 = 44 bytes
Extended I/O fault table	81 = 48 bytes

Example 1:

In the following example, when input %I00251 is on and input %I00250 transitions on, the last entry in the PLC fault table is read into the parameter block. When input %I00251 is off and input %I00250 transitions on, the last entry in the I/O fault table is read into the parameter block. The parameter block is located in global memory at location %P00600.



Example 2:

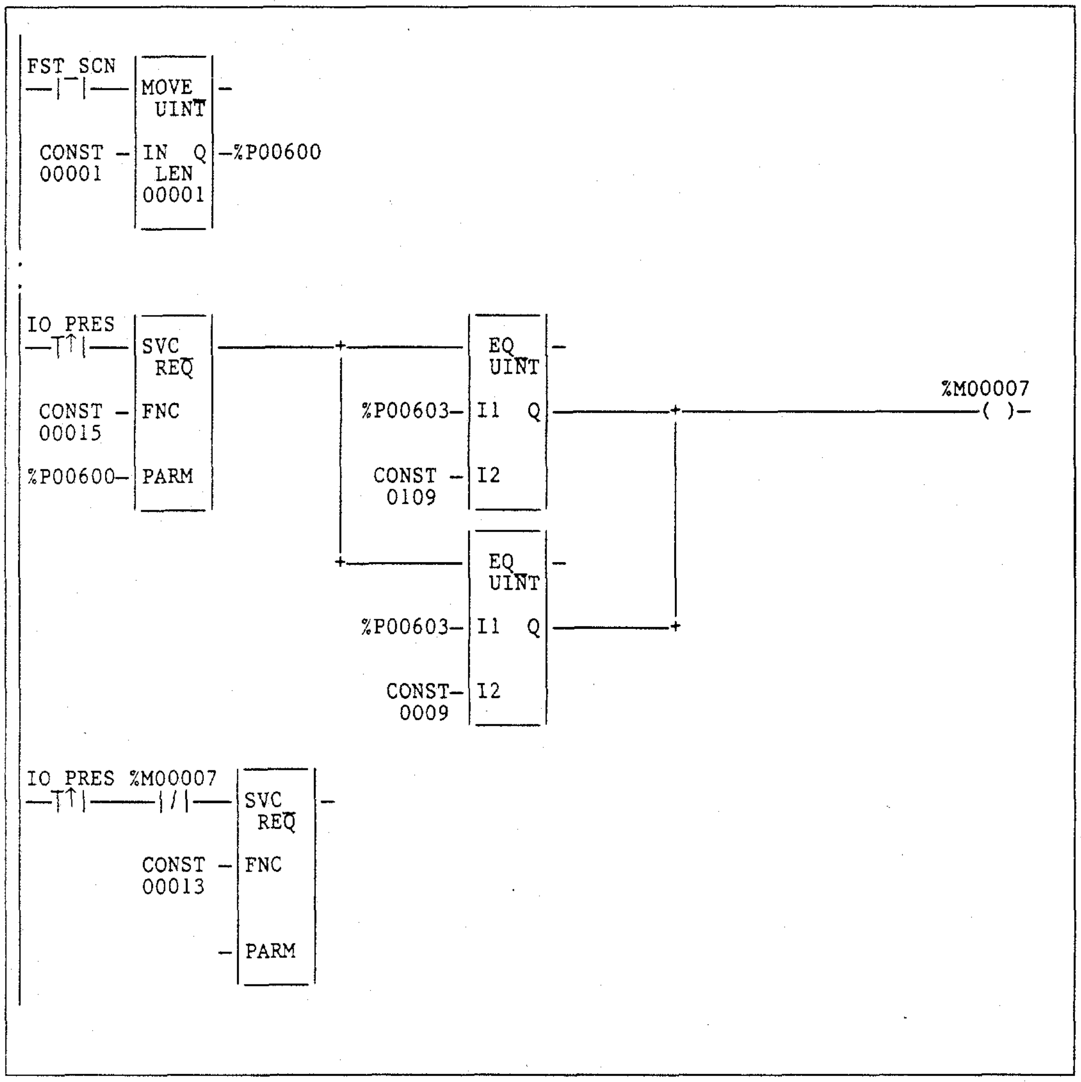
In the next example, the PLC is shut down when any fault occurs on an I/O module except when the fault occurs on boards in rack 0, slot 9 and in rack 1, slot 9. If faults occur on these two modules, the system remains running. The parameter for "table type" is set up on the first sweep. The contact IO_PRES, when set, indicates that the I/O fault table contains an entry. The PLC CPU sets the up transition contact in the sweep after the fault logic places a fault in the table. If faults are placed in the table in two consecutive sweeps, the up transition contact is set for two consecutive sweeps.

The example uses a parameter block located at global memory %P00600. After the SVCREQ function executes, the fourth, fifth, and sixth words of the parameter block contain the address of the I/O module that faulted:

High Byte	Low Byte	
	1	%P00600
	long/short	%P00601
	reference address	%P00602
slot number	rack number	%P00603
bus address	I/O bus no.	%P00604
	point address	%P00605

fault data

In the program, the EQ_UINT blocks compare the rack/slot address in the table to hexadecimal constants. The internal coil %M00007 is turned on when the rack/slot where the fault occurred meets the criteria specified above. If %M00007 is on, its normally closed contact is off, preventing the shutdown. Conversely, if %M00007 is off because the fault occurred on a different module, the normally closed contact is on and the shutdown occurs.



SVCREQ #16: Read Elapsed Time Clock

Use SVCREQ function #16 to read the value of the system's elapsed time clock. This clock tracks elapsed time in seconds since the PLC powered on. The timer will roll over approximately once every 100 years.

The parameter block is an output parameter block only; it has a length of three words.

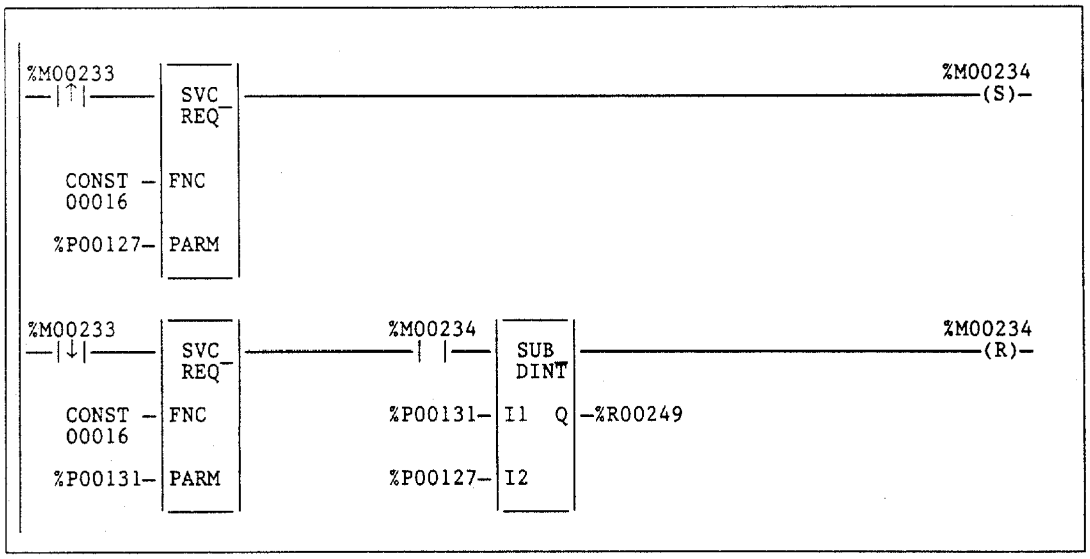
seconds from power on (low order)	address
seconds from power on (high order)	address + 1
100 microsecond ticks	address + 2

The first two words are the elapsed time in seconds. The last word is the number of 100 microsecond ticks in the current second.

Example:

In the following example, when internal coil %M00233 transitions on, the value of the elapsed time clock is read and coil %M00234 is set. When it transitions off, the value is read again. After the off transition, the difference between the values is calculated and the result is stored in register memory at location %R00250.

The parameter block for the first read is at %P00127; for the second read, at %P00131. The calculation ignores the number of hundred microsecond ticks and the fact that the DINT type is actually a signed value. The calculation is correct until the time since power on reaches approximately 50 years.



SVCREQ #17: Mask/Unmask I/O Interrupt

Use SVCREQ function #17 to mask or unmask an interrupt from an input board. When an interrupt is masked, the PLC CPU will not execute the corresponding interrupt block when the input transitions and causes an interrupt.

The parameter block is an input parameter block only; it has a length of three words.

0 = unmask input 1 = mask input	address
memory type	address + 1
reference (offset)	address + 2

"Memory type" is a decimal number that resides in the low byte of word address + 1. It corresponds to the memory type of the input:

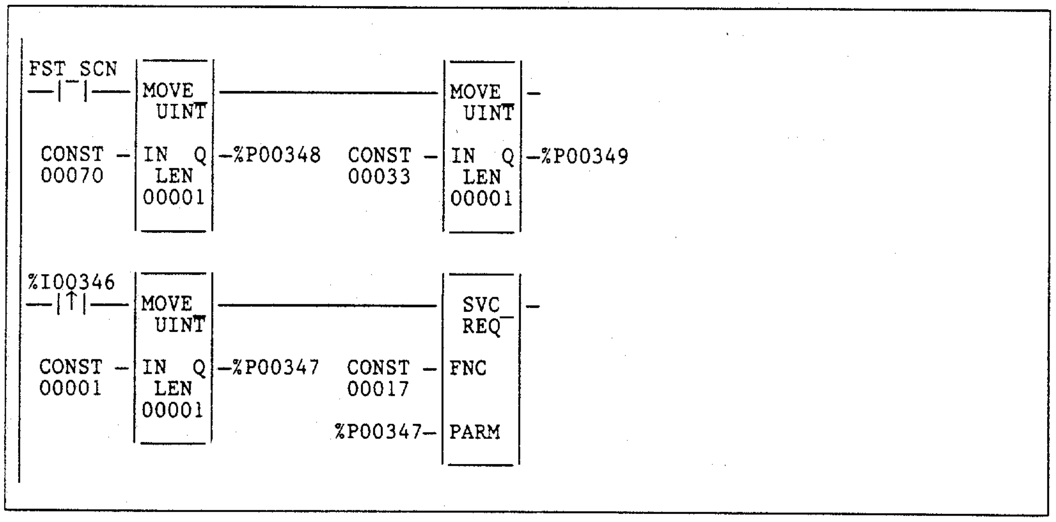
```
70 = %I memory in bit mode
10 = %AI memory
```

Successful execution will occur unless:

- Some number other than 0 or 1 is entered as the requested operation.
- The memory type of the input to be masked or unmasked is not %I or %AI memory.
- The I/O board is not an appropriate Series 90 input module.
- The reference address specified does not correspond to a valid interrupt trigger reference.
- The specified channel does not have its interrupt enabled in the configuration.

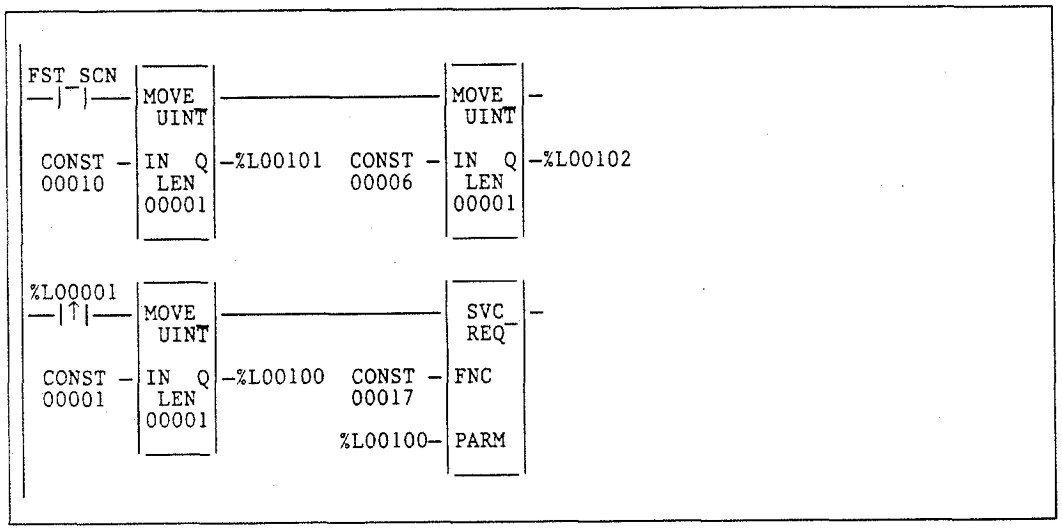
Example 1:

In the following example, when %I00346 transitions on, interrupts from input %I00033 are masked. The parameter block at %P00347 is set up on the first sweep.



Example 2:

In the following example, when %L00001 transitions on, alarm interrupts from input %AI0006 are masked. The parameter block at %L00100 is set up on the first sweep.



SVCREQ #18: Read I/O Override Status

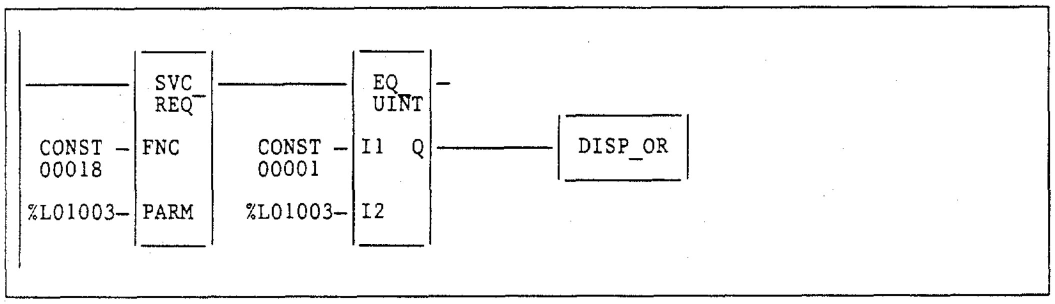
Use SVCREQ function #18 in order to read the current status of %I and %Q overrides in the CPU.

The parameter block is an output parameter block only; it has a length of one word.

0 = No overrides are set.	address
1 = Overrides are set.	

Example:

In the following example, the status of I/O overrides is always read into location %L01003. If any overrides are present, program block DISP_OR is called.



Note

SVCREQ #18 does not detect overrides in %G or %M memory types. Use %S0011 (OVR_PRE) to detect overrides in %I, %Q, %G, and/or %M memory types..

SVCREQ #19: Set Run Enable/Disable

Use SVCREQ function #19 to permit the ladder program to control the **RUN** mode of the CPU.

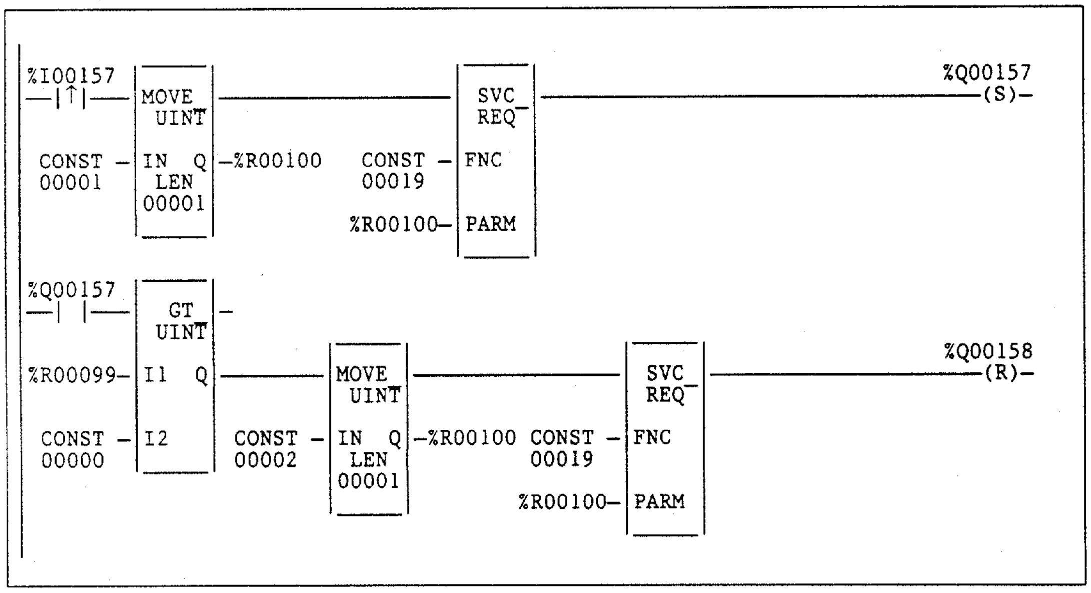
The parameter passed indicates which function to perform. The OK output is turned ON if the function executes successfully. It is set OFF if the requested operation is not **SET RUN DISABLE** mode (1) or **SET RUN ENABLE** mode (2).

The parameter block is an input parameter block only with this format:

Address	1 = **SET RUN DISABLE** mode. 2 = **SET RUN ENABLE** mode.

Example:

In the following example, when input %I00157 transitions to on, the **RUN DISABLE** mode is set. When the SVCREQ function successfully executes, coil %Q00157 is turned on. When %Q00157 is on and register %R00099 is greater than zero, the mode is changed to **RUN ENABLE** mode. When the SVCREQ successfully executes, coil %Q00157 is turned off.



SVCREQ #20: Read Fault Tables

Use SVCREQ function #20 to retrieve the entire PLC or I/O fault table and return it to the ladder program in designated registers. The first input parameter designates which table is to be read. A second input parameter (always zero for the standard Read Fault Tables) is used by the extended format to read a designated fault entry or to read a range of fault entries. The fault table data is placed in the parameter block following the input parameters.

The OK output is turned on if the function executes successfully. It is off if the requested operation is not Read PLC Fault Table (00h), Read I/O Fault Table (01h), Read Extended PLC Fault Table (80h), or Read Extended I/O Fault Table (81h), or if there is not enough of the specified memory reference to hold the fault data. If the specified fault table is empty, the function sets the OK output on but returns only the fault table header information.

Input and Output Parameter Format for the Non-Extended Formats

The parameter block is an input and output parameter block. For Read PLC Fault Table (00h) and Read I/O Fault Table (01h), the input parameter block has the following format:

Address	00h = Read PLC fault table. 01h = Read I/O fault table.
Address + 1	Always zero (0).

The output parameter block has this format:

Address	0 = PLC fault table 1 = I/O fault table
Address + 1	Always zero (0)
Address + 2 thru Address + 14	Unused
Address + 15 Address + 16 Address + 17	Time since last clear
Address + 18	Number of faults since last clear
Address + 19	Number of faults in queue
Address + 20	Number of faults read
Address + 21	Start of fault data

For the non-extended formats, each fault table entry is 21 words long (42 bytes). There are a maximum of 16 PLC fault table entries and 32 I/O fault table entries. If the fault table read is empty, no data is returned.

Note

SVCREQ #20 will not work unless there are 693 registers available.

Input and Output Parameter Format for the Extended Formats

The parameter block is an input and output parameter block. For Read Extended PLC Fault Table (80h) and Read Extended I/O Fault Table (81h), the input parameter block has the following format:

Address	80h = Read extended PLC fault table. 81h = Read extended I/O fault table.
Address + 1	Starting index of faults to be read.
Address + 2	Number of faults to be read.

The output parameter block has this format:

Address	80h = Extended PLC fault table 81h = Extended I/O fault table
Address + 1	Starting index of faults to be read.
Address + 2 thru Address + 14	Unused
Address + 15 Address + 16 Address + 17	Time since last clear
Address + 18	Number of faults since last clear
Address + 19	Number of faults in queue
Address + 20	Number of faults read
Address + 21	
Address + 22	
Address + 23	
Address + 24	
Address + 25	
Address + 26	
Address + 27	PLC name
Address + 28	
Address + 29	
Address + 30	
Address + 31	
Address + 32	
Address + 33	
Address + 34	
Address + 35	
Address + 36	
Address + 37	Start of fault data

For Read Extended PLC Fault Table (80h) and Read Extended I/O Fault Table (81h), each extended fault table entry is 23 words long (46 bytes). There are a maximum of 40 PLC fault table entries and 40 I/O fault table entries. The default values are 16 PLC fault table entries and 32 I/O fault table entries. If the fault table read is empty, no data is returned.

Note

For the non-extended format, SVCREQ #20 will not work unless there are 693 consecutive registers available (beginning with the starting point). For the extended format, SVCREQ #20 will not work unless there are 958 consecutive registers available (beginning with the starting point).

The total size of the fault table for the extended fault format is
Header Size + ((# fault entries) * (size of fault entry))

The following table shows the return format of both a PLC fault table entry and an I/O fault table entry in non-extended format.

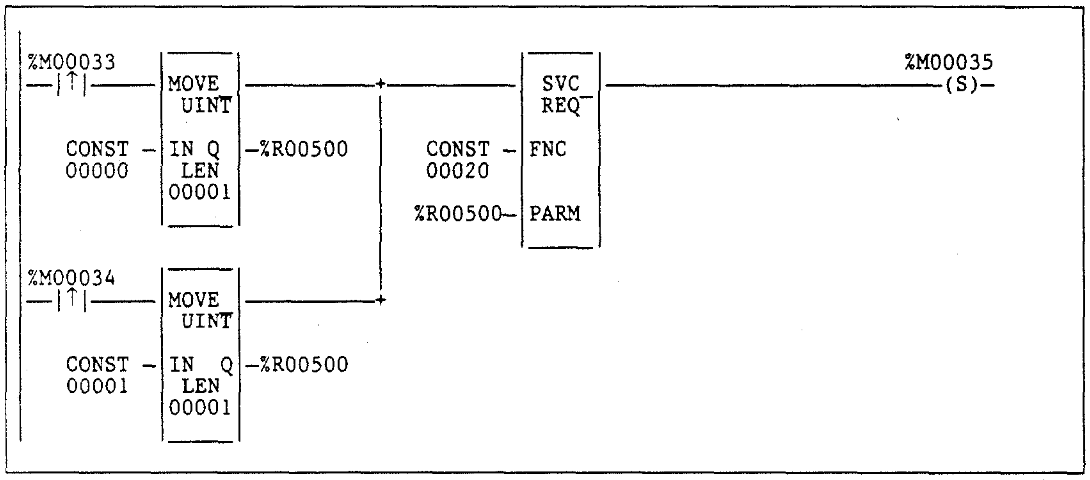
Address	PLC Fault Table	I/O Fault Table
Address + 21	Long/Short	Long/Short
Address + 22	Spare	Reference Address
Address + 23	PLC fault address	I/O fault address
Address + 24	PLC fault address	I/O fault address
Address + 25	Fault group and action	I/O fault address
Address + 26	Error code	Fault group and action
Address + 27	Fault extra data	Fault category and type
Address + 28	Fault extra data	Fault description
Address + 29	Fault extra data	Fault specific data
Address + 30	Fault extra data	Fault specific data
Address + 31	Fault extra data	Fault specific data
Address + 32	Fault extra data	Fault specific data
Address + 33	Fault extra data	Fault specific data
Address + 34	Fault extra data	Fault specific data
Address + 35	Fault extra data	Fault specific data
Address + 36	Fault extra data	Fault specific data
Address + 37	Fault extra data	Fault specific data
Address + 38	Fault extra data	Fault specific data
Address + 39	Time stamp	Time stamp
Address + 40	Time stamp	Time stamp
Address + 41	Time stamp	Time stamp

The Long/Short indicator in the first byte of Address + 21 defines the quantity of fault data present in the fault entry. In the PLC fault table, a long/short value of 00 represents 8 bytes of fault extra data present in the fault entry, and 01 represents 24 bytes of fault extra data. In the I/O fault table, 02 represents 5 bytes of fault specific data, and 03 represents 21 bytes.

For an explanation of each field, refer to Appendix B, "Interpreting Fault Tables Using Logicmaster 90-70 Software."

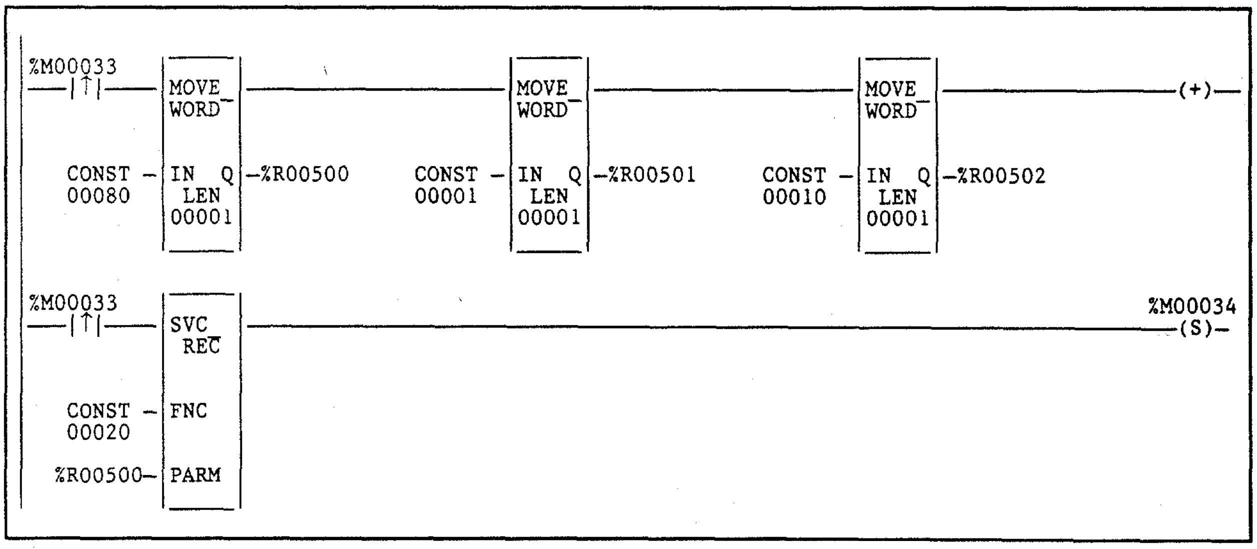
Example (Non-Extended Format):

In the example below, when input %M00033 transitions on, the PLC fault table is read. When %M00034 transitions on, the I/O fault table is read. The parameter block is located at %R00500. When the SVCREQ function successfully executes, coil %M00035 is turned on.



Example Extended Format):

In the example below, when input %M00033 transitions on, the Extended PLC fault table is read. The parameter block is located at %R00500. %R00500 contains the fault table type (PLC Extended); %R00501 contains the starting fault to read, and %R005002 contains the number of faults to read starting with the fault number in %R00501. When the SVCREQ function successfully executes, coil %M00034 is turned on.



SVCREQ #21: User-Defined Fault Logging

Use SVCREQ function #21 to define a fault that can be displayed in the PLC fault table. The fault contains binary information or an ASCII message. The user-defined fault codes start at 0 hex.

The error code information for the fault must be within the range 0 to 2047 in order for an "Application Msg:" to be displayed. If the error code is in the range 81 to 112 decimal the PLC CPU sets a fault bit of the same number in %SA system memory. This allows up to 32 bits to be individually set.

Error Code	Status Bit
Error 0 - 80	No bit set.
Error 81 - 112	Sets %SA.
Error 113 - 2047	No bit set.
Error 2048 – 32,767	Reserved.

When EN is active, the fault data array referenced by IN is logged as a fault to the PLC fault table. If EN is not enabled, the ok bit is cleared. If the error code is out of range, the ok bit is cleared and the fault will not be logged as requested.

The parameter block is an input parameter block only with this format:

Parameter	MSB	LSB
Address	Error Code	
Address + 1	Text 2	Text 1
Address + 2	Text 4	Text 3
Address + 3	Text 6	Text 5
Address + 4	Text 8	Text 7
Address + 5	Text 10	Text 9
Address + 6	Text 12	Text 11
Address + 7	Text 14	Text 13
Address + 8	Text 16	Text 15
Address + 9	Text 18	Text 17
Address + 10	Text 20	Text 19
Address + 11	Text 22	Text 21
Address + 12	Text 24	Text 23

The input parameter data allows you to select an error code in the range 0 to 2047 and text information that will be placed in the fault extra data portion of a long PLC fault. The PLC fault address, fault group, and fault action are filled in by the function block.

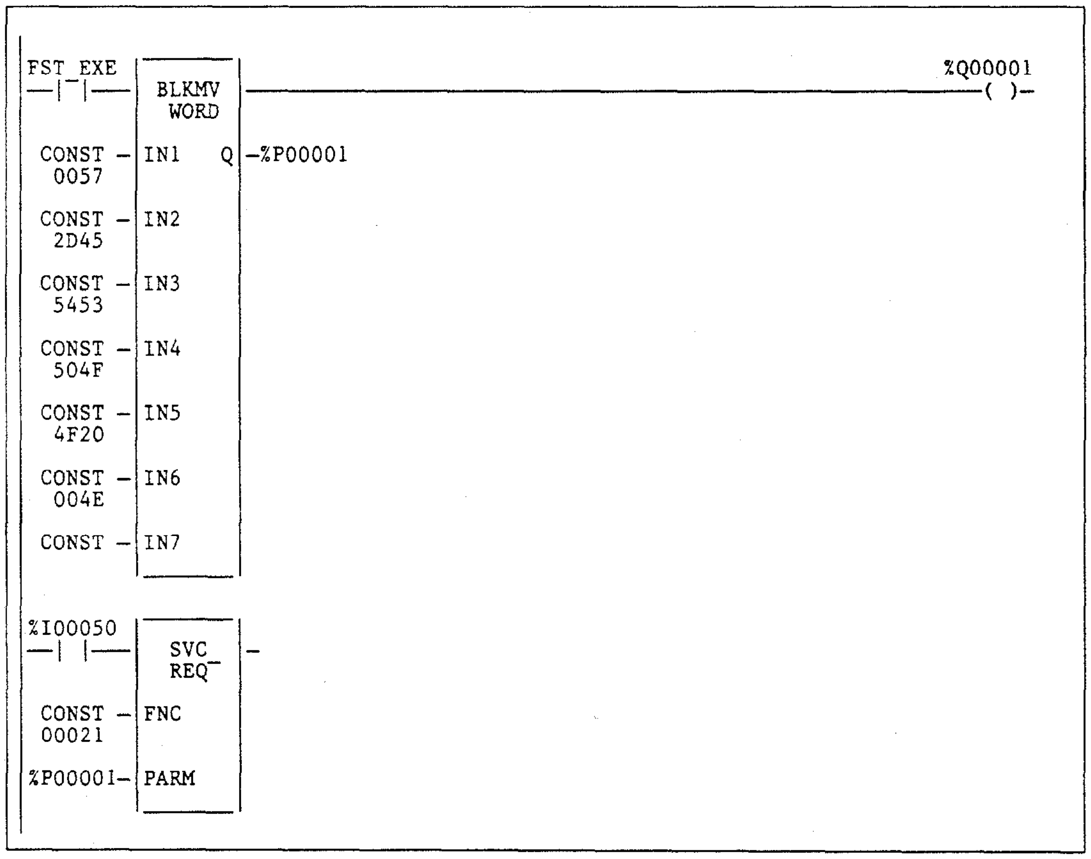
The fault text bytes 1 – 24 can be used to pass binary or ASCII data with the fault. If the first byte of the fault text data is non-zero, the data will be an ASCII message string. This message will then be displayed in the fault description area of the fault table. If the message is less than 24 characters, the ASCII string must be NULL byte-terminated. The programmer will display "Application Msg:" and the ASCII data will be displayed as a message immediately following "Application Msg:". If the error code is between 1 and 2047 the error code number will be displayed immediately after "Msg" in the "Application Msg:" string. If the error code is greater than 2047, it will be converted to error code 0.

If the first byte of text is zero, then only "Application Msg:" will display in the fault description. The next 1-23 bytes will be considered binary data for user data logging. This data can be displayed by using the CTRL-F display of fault data in the Logicmaster 90-70 programmer PLC fault display.

For more information, refer to chapter 5, "PLC Control and Status," in the *Programming Software User's Manual*, GFK-0263.

Example:

In the following example, the value passed to IN1 is the fault error code. The value passed in, 16x0057, represents an error code of 87 and will appear as part of the fault message. The values of the next inputs give the ASCII codes for the text of the error message. For IN2, the input is 2D45. The low byte, 45, decodes to the letter **E** and the high byte, 2D, decodes to _. Continuing in this manner, the string continues with **S T O P O** and **N**. The final character, **00**, is the null character which terminates the string. Thus, the decoding yields the string message **E_STOP ON**.



SVCREQ #22: Mask/Unmask Timed Interrupts

Use SVCREQ function #22 to mask or unmask timed interrupts and to read the current mask. When the interrupts are masked, the PLC CPU will not execute any interrupt block that is associated with a timed interrupt. Timed interrupts are masked/unmasked as a group. They cannot be individually masked or unmasked.

Successful execution will occur unless some number other than 0 or 1 is entered as the requested operation or mask value.

The parameter block is an input and output parameter block.

To determine the current mask, use this format:

0 = Read interrupt mask.	address

The PLC returns this format:

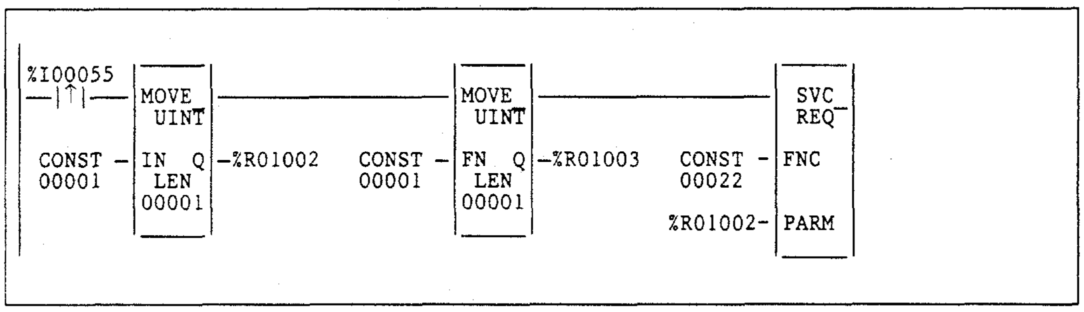
0 = Read interrupt mask.	address
0 = Timed interrupts are unmasked. 1 = Timed interrupts are masked.	address + 1

To change the current mask, use this format:

1 = Mask/unmask interrupts.	address
0 = Unmask timed interrupts. 1 = Mask timed interrupts.	address + 1

Example:

In the following example, when input %I00055 transitions on, timed interrupts are masked.



SVCREQ #23: Read Master Checksum

SVCREQ function #23 returns master checksums for the set of user program(s) and the configuration. It also returns the checksum for the block from which the service request is made.

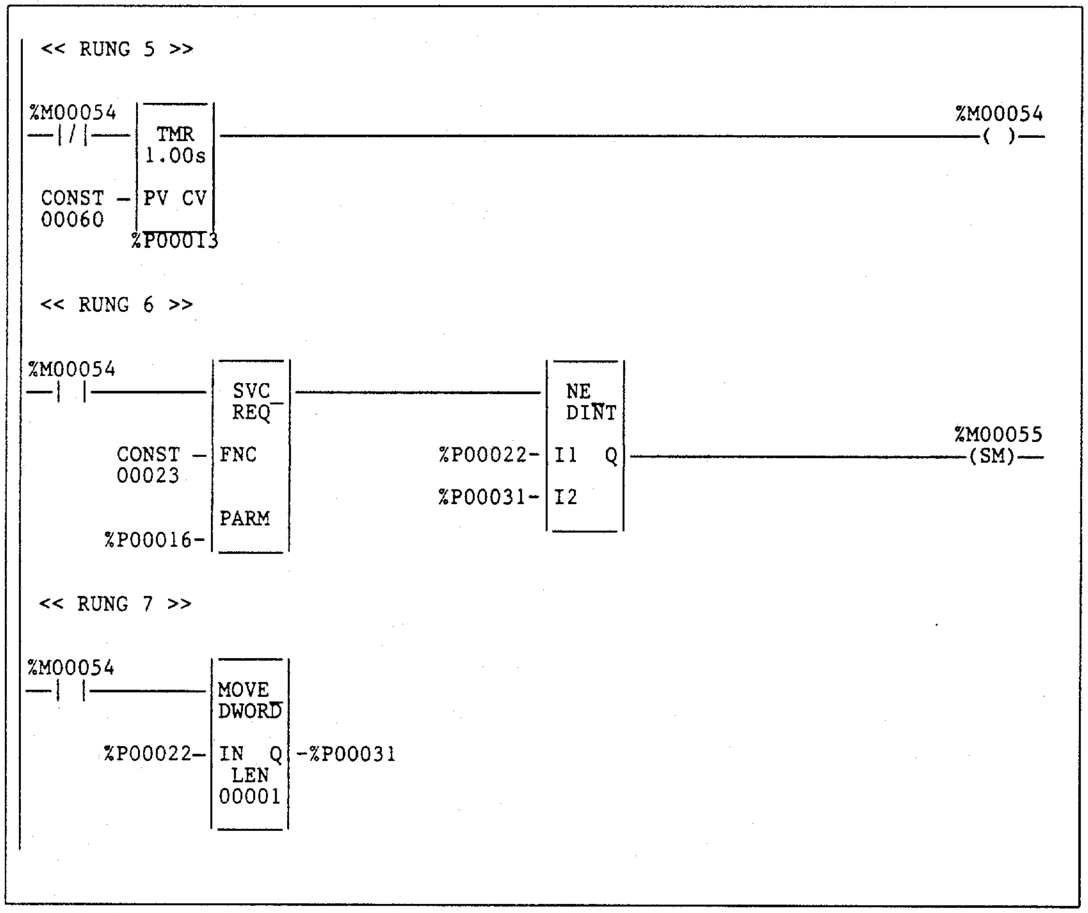
When a **RUN MODE STORE** is active, the program checksums may not be valid until the store is complete. To determine when checksums are valid, three flags (one each for Program Block Checksum, Master Program Checksum, and Master Configuration Checksum) are provided at the beginning of the output parameter block.

There is no input parameter block for this service request. The output parameter block layout is as follows; it requires 15 words of memory.

Output Parameter Block		**Word Address**
Program Checksum Valid (0 = not valid, 1 = valid)		address
Master Program Checksum Valid (0 = not valid, 1 = valid)		address + 1
Master Configuration Checksum Valid (0 = not valid, 1 = valid)		address + 2
Number of LD/SFC Blocks (including _MAIN)		address + 3
Size of User Program in bytes (DWORD data type)		address + 4
Program Set Additive Checksum		address + 6
Program CRC Checksum (DWORD data type)		address + 7
Size of Configuration Data in Bytes		address + 9
Configuration Additive Checksum		address + 10
Configuration CRC Checksum (DWORD data type)		address + 11
Always zero (high byte)	Currently Executing Block's Additive Checksum	address + 13
Currently Executing Block's CRC Checksum		address + 14

Example:

In the following example, when the timer using registers %P00013 through %P00015 expires, the checksum read is performed. The checksum data returns in registers %P00016 through %P00030. The master program checksum in registers %P00022 and %P00023 (the program checksum is a DWORD data type and occupies two adjacent registers) is compared with the last saved master program checksum. If these are different, coil %M00055 is latched on. The current master program checksum is then saved in registers %P00031 and %P00032.



SVCREQ #25: Disable/Enable EXE Block and Standalone C Program Checksums

Use SVCREQ function #25 to enable or disable the inclusion of EXE blocks and standalone C programs (i.e., C applications) in the background checksum calculation. The default is to include the checksums.

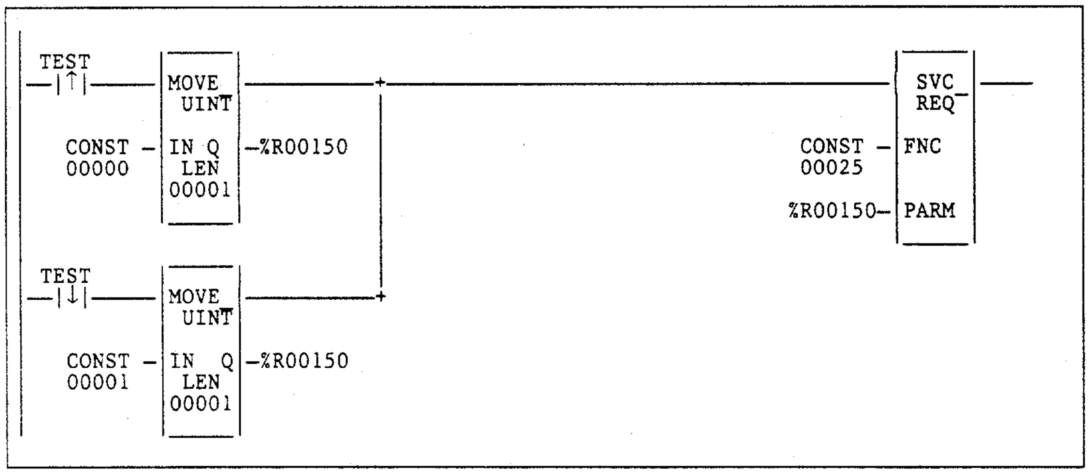
This service request uses only an input parameter block.

0 = Disable C applications inclusion in checksum calculation. 1 = Enable C applications inclusion in checksum calculation.	address

The parameter block is unchanged after execution of the service request.

Example:

In the following example, when the coil TEST transitions from OFF to ON, SVCREQ #25 executes to disable the inclusion of C applications in the background checksum calculation. When coil TEST transitions from ON to OFF, the SVCREQ executes to again include C applications in the background checksum calculation.



SVCREQ #26: Role Switch

Note

SVCREQ #26 is intended for use with Hot Standby CPU Redundancy which is only available on Model 780 CPUs (IC697CPU780). For more information about Hot Standby CPU Redundancy, refer to the *Series 90™-70 Hot Standby CPU Redundancy User's Guide* (GFK-0827).

Use SVCREQ function #26 to cause the CPUs to switch roles on the next sweep (active to backup and backup to active) if the CPUs are synchronized and the timing requirements of the role switch request are met. A manual role switch cannot occur within 10 seconds of a previous manual role switch. Role switches due to failures or resynchronization are always allowed (the 10 second limitation does not apply).

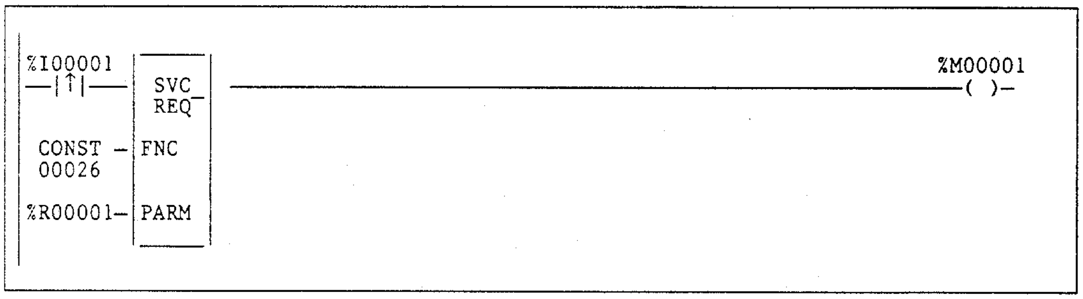
Note

Power flow from SVCREQ #26 indicates that a role switch will be attempted on the next sweep. It does **not** indicate that a role switch has occurred or that a role switch will occur on the next sweep.

This function has no associated parameter block; however, the programming software requires that an entry be made for PARM. Enter any appropriate reference here; it will not be used.

Example:

In the following example, a switch on a control console is wired to %I00001, the input to the SVCREQ #26 function block. When closed, the switch will activate the SVCREQ #26, causing a role switch between CPUs.



The 10-second limitation allows this SVCREQ to be in both CPUs so that only a single switch occurs if the input is seen by both CPUs.

SVCREQ #27 and #28: Write to/Read from Reverse Transfer Area

Note

SVCREQ #27 and #28 are intended for use with Hot Standby CPU Redundancy which is only available on Model 780 CPUs (IC697CPU780). For more information about Hot Standby CPU Redundancy, refer to the *Series 90™-70 Hot Standby CPU Redundancy User's Guide* (GFK-0827).

SVCREQ functions #27 and #28 enable you to transfer 8 bytes (4 registers) of data from a backup CPU to the active CPU. SVCREQ #27 is used to initiate this transfer of data by copying the eight bytes of data from the reference specified by PARM to a temporary buffer located on the active CPU. SVCREQ #28 (Read from Reverse Transfer Area) is then executed to copy the eight bytes of data from the temporary buffer on the active CPU to the reference specified by PARM.

Note

There is a one-sweep delay between sending the data to the active CPU and reading the data on the active CPU.

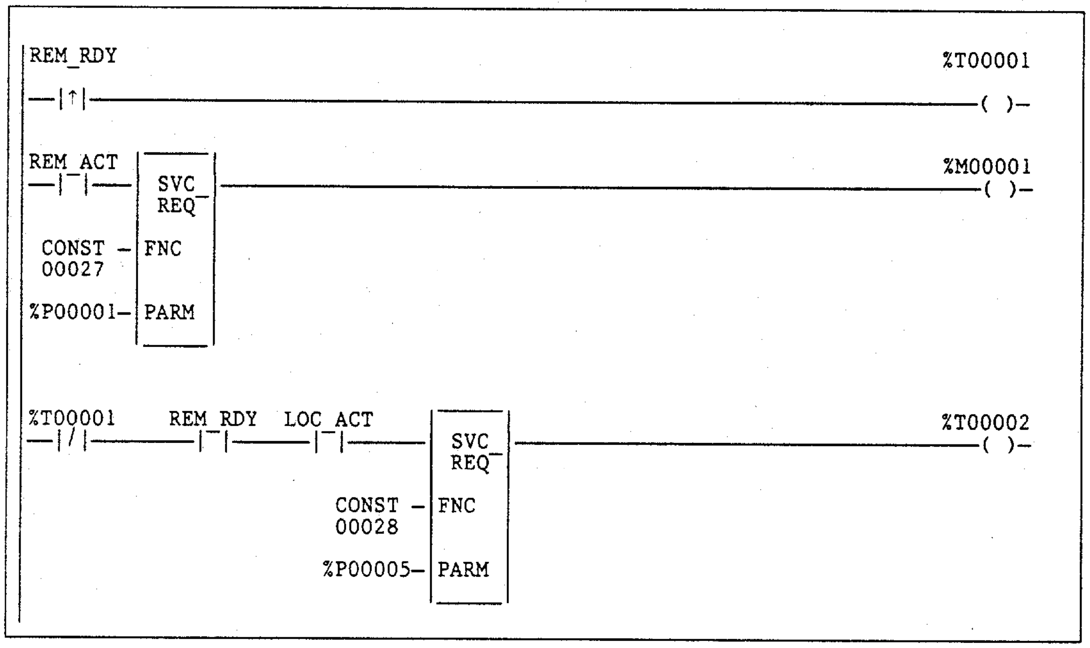
The data copied from the buffer is not valid:

- During the first scan after either CPU has transitioned to **RUN** mode.
- While the backup CPU is in **STOP** mode.
- If the remote CPU does not issue a service request to the sending CPU.

Example:

In the following example, two rungs are placed in the program logic of both CPUs. The backup CPU will send %P0001 through %P0004 to the active CPU. The active CPU will read the data into %P0005 through %P0008. %P0001 through %P0004 on the active CPU and %P0005 through %P0008 on the backup CPU will not change. %T0002 will be set whenever the operation is successful and the data can be used.

The data should not be used if REM_RDY is off or if REM_RDY is transitioning to on.


REM_RDY
%T00001
REM_ACT
SVC_REQ
%M00001
CONST 00027
FNC
%P00001
PARM
%T00001
REM_RDY
LOC_ACT
SVC_REQ
%T00002
CONST 00028
FNC
%P00005
PARM

SVCREQ #32: Suspend/Resume I/O Interrupt

Use SVCREQ function #32 to suspend a set of I/O interrupts and cause occurrences of these interrupts to be enqueued until these interrupts are resumed. The set of I/O interrupts are those that can be generated from the 90-70 High Speed Counter. The number of I/O interrupts that can be enqueued depends on the I/O module's capabilities. The PLC CPU informs the I/O module that its interrupts are to be suspended or resumed. The I/O module's default is resumed. The suspend applies to all I/O interrupts associated with the I/O module. Interrupts should be suspended and resumed within a single sweep.

This service request uses only an input parameter block. It's length is three words.

0 = resume interrupt. 1 = suspend interrupt.	address
memory type	address + 1
reference (offset)	address + 2

Successful execution will occur unless:

- Some number other than 0 or 1 is passed in as the first parameter.
- The memory type parameter is not 70 (%I memory).
- The I/O module associated with the specified address is not an appropriate module for this operation. (The module must be a 90-70 High Speed Counter.)
- The reference address specified is not the first %I reference for the High Speed Counter.
- Communication between the PLC CPU and this I/O module has failed. (The board is not present, or it has experienced a fatal fault.)

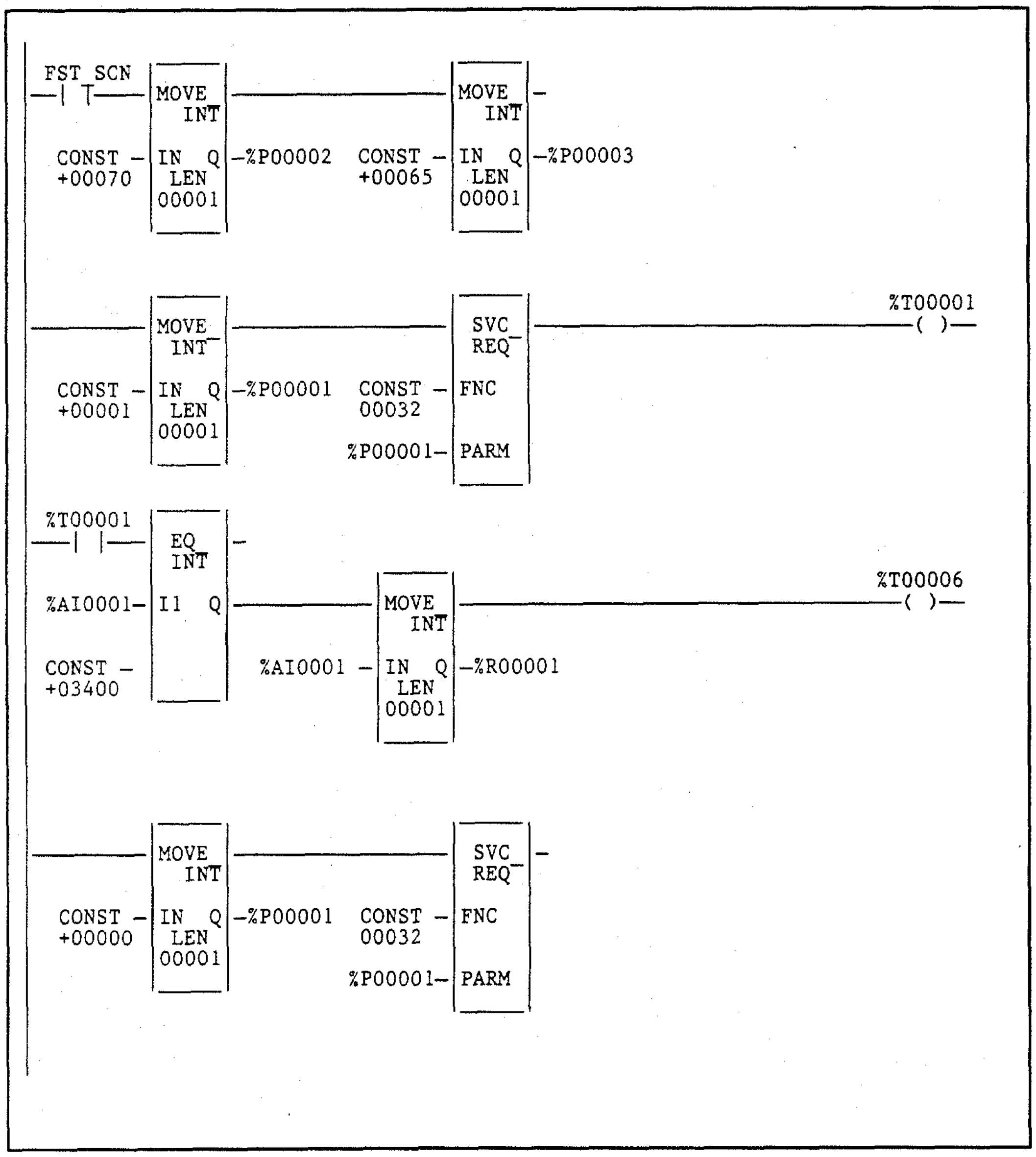
Example:

In the following example, interrupts from the high speed counter module whose starting point reference address is %I00065 will be suspended while the CPU solves the logic of the second rung. Without the Suspend, an interrupt from the HSC could occur during execution of the third rung, and %T00006 could be set while %R000001 has a value other than 3400. (%AI00001 is the first nondiscrete input reference for the High Speed Counter.)

Note

I/O interrupts, unless suspended or masked, can interrupt the execution of a function block. The most often used application of this Service Request is to prevent the effects of the interrupts for diagnostic or other purposes.

Example



PID

The PID function is designed to solve one loop equation in one execution. The function block data uses 40 registers in a loop data table. The first 35 registers are reserved for the function and should not be used by any application program. The last 5 registers are reserved for external use.

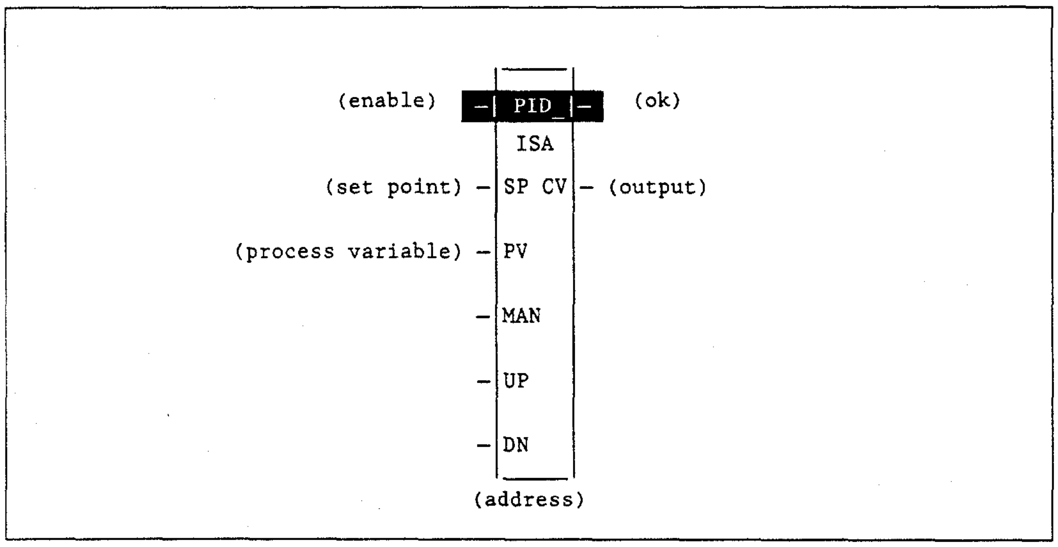
Registers cannot be shared. If there are multiple occurrences of the same PID function controlling multiple loops, each occurrence requires a separate block of 40 registers.

The PIDISA and PIDIND functions provide two PID (proportional/integral/derivative) closed-loop control algorithms.

The PID function has seven input parameters: a boolean enable (EN), a process set point (SP), a process variable (PV), a manual/auto boolean switch (MAN), a manual mode up adjustment input (UP), and a manual mode down adjustment (DN). It also has an address, which specifies the location of a block of parameters associated with the function. It has two output parameters, a successful boolean output (ok) and the control variable result (CV).

When there is power flow at EN and no power flow at MAN, the PID algorithm is applied to SP and PV, with the result placed in CV. OK is set ON if the PID function executes successfully; otherwise, it is set OFF.

When there is power flow at EN and MAN, the PID block is placed into **MANUAL** mode. The CV output maintains its current value and can be adjusted with the UP and DN inputs. While the PID block is in **MANUAL** mode, the PID algorithm is executed so that the calculated result tracks with the manually controlled CV value. This prevents the PID function from building up an integral component while in **MANUAL** mode, and provides bumpless transfer when the block is placed back into **AUTOMATIC** mode.



Parameters:

Parameter	Description
enable	When enabled, the PID function is performed.
SP	SP is the control loop set point.
PV	PV is the control loop process variable.
MAN	When energized, the PID function is in **MANUAL** mode.
UP	When energized, if in **MANUAL** mode, the CV output is adjusted up.
DN	When energized, if in **MANUAL** mode, the CV output is adjusted down.
address	Address is the location of the PID control block information, which consists of 40 consecutive registers of %R, %P, or %L memory. Note: Do not use this address with other instructions. Caution: Overlapping references will result in erratic operation of the PID algorithm.
ok	The ok output is energized when the function is performed without error.
CV	CV is the control variable output.

Valid Memory Types:

Parameter	flow	%I	%Q	%M	%T	%S	%G	%U	%R	%P	%L	%AI	%AQ	%UR	const	none
enable	•															
SP	•	•	•	•	•		•		•	•	•	•	•	•	•	
PV	•	•	•	•	•		•		•	•	•	•	•	•		
MAN	•															•
UP	•															
DN	•															•
address									•	•	•			•		
ok	•															•
CV	•	•	•	•	•		•		•	•	•	•	•	•		

• Valid reference or place where power may flow through the function.

Note

For restrictions within a Parameterized Subroutine Block, see page 2-34.

PID Function Block Data:

The PID function block contains the following data items. %Ref is the starting reference address assigned to the PID function block in the location field.

%Ref+0000	Loop Number *
%Ref+0001	Algorithm **
%Ref+0002	Sample Period *
%Ref+0003	Dead Band + *
%Ref+0004	Dead Band – *
%Ref+0005	Proportional Gain *
%Ref+0006	Derivative *
%Ref+0007	Integral Rate *
%Ref+0008	Bias *
%Ref+0009	Upper Clamp *
%Ref+0010	Lower Clamp *
%Ref+0011	Minimum Slew Time *
%Ref+0012	Config Word *
%Ref+0013	Manual Command *
%Ref+0014	Control Word **
%Ref+0015	Internal SP **
%Ref+0016	Internal CV **
%Ref+0017	Internal PV **
%Ref+0018	Output **
%Ref+0019	Diff Term Storage **
%Ref+0020	Int Term Storage **
%Ref+0021	Int Term Storage **
%Ref+0022	Slew Term Storage **
%Ref+0023 %Ref+0024 %Ref+0025	Clock ** (time last executed)
%Ref+0026	Y Remainder Storage **
%Ref+0027	Lower Range for SP, PV *
%Ref+0028	Upper Range for SP, PV *
%Ref+0029 • %Ref+0034	Reserved for internal use
%Ref+0035 • %Ref+0039	Reserved for external use

*	=	May be set by the user.
**	=	Set and maintained by the PLC.

The loop number, execution interval, deadband +/−, proportional gain, differential gain, integral rate, bias, upper/lower clamp, minimum slew time, and config word values must be set by the application program. The other values are maintained by the PID function block.

Table 4-4. PID Function Block Data

Data Item	Description
Loop Number	An unsigned integer that provides a common identification in the PLC with the loop number defined by an operator interface device. The loop number is displayed under the block address when logic is monitored from the Logicmaster 90 software. Use of the loop number is optional.
Algorithm	An unsigned integer that is set by the PLC to identify what algorithm is being used by the function block. The ISA algorithm is defined as algorithm 1, and the interactive algorithm is identified as algorithm 2.
Sample Period	The time in increments of 0.01 seconds between executions of the function block. The PID function is calculated at this interval. The function compensates for the actual time elapsed since the last execution, within 100 microseconds. If this value is set to zero, the function is executed each time it is enabled.
Dead Band (+/—)	Signed word values defining the upper (+) and lower (−) limit of the dead band interval, in counts. If no dead band is required, these terms should be set to zero. If the error is between the dead band (+) and (−) values, the function is solved with the error term set to zero. In other words, the error must grow beyond these limits before the PID block will begin to adjust the CV output in response.
Proportional Gain	A signed word value that sets the proportional gain, in hundredths.
Derivative	A signed word value that sets the derivative, in hundredths of seconds.
Integral Rate	A signed word value that sets the integral rate, in units of repeats per 1000 seconds.
Bias	A signed word value that sets the bias term, in units of counts. Feed-forward control can be implemented by adjusting this value.
Upper and Lower Clamps	Signed word values that define the upper and lower limits on the CV output, in units of counts. Anti-reset windup is applied to the PID integral term when a clamp limit is reached. The integral term is adjusted to a value that holds the output at the clamped value.
Minimum Slew Time	An unsigned word value that defines the output minimum slew time. This term limits how quickly the output is allowed to change from 0 to 100%. This has the effect of limiting how quickly the integral term is allowed to change, preventing windup. If no slew rate limit is desired, this term should be set to zero. The slew rate limit is given in seconds for full travel.

Table 4-4. PID Function Block Data (cont'd)

Data Item	Description
Config Word	A word value with the following format: **0** = Error Term. When this bit is set to zero, the error term is SP - PV. When this bit is set to 1, the error term is PV - SP. **1** = Output Polarity. When this bit is set to zero, the CV output represents the output of the PID calculation. When it is set to 1, the CV output represents the negative of the output of the PID calculation. **2** = Derivative action on PV. When this bit is set to zero, the derivative action is applied to the error term. When it is set to 1, the derivative action is applied to PV. All remaining bits should be zero.
Manual Command	A signed word value that defines the output when in **MANUAL** mode.
Control Word	A discrete data structure with the following format: **0** = Override. **1** = Auto/Manual. **2** = Enable. **3** = Raise. **4** = Lower. **Override**: When the override bit is set to 1, the function block is executed based upon the current values of up, down, and manual; these values will not be written with the discrete inputs into the function block. When the override bit is set to 0, the up, down, and manual values are set to the values, as defined by the function block discrete inputs. Override also affects the values used for SP. If override is set, the function block will not update the value of SP and will execute based upon the SP value in the data structure. The purpose of the override bit is to allow the operator interface device to take control of the boolean inputs into the function block so that they may be controlled by the operator interface device. In addition, since SP is not updated, the operator interface unit can also set override and take control of the set point. **Enable**: The enable bit will track the enable input into the function block. **Manual/Raise/Lower**: These three bits represent the state of the three boolean inputs into the function block when the override bit is 0. Otherwise, they can be manipulated by an outside source.
SP	This is a signed word value representing the set point input to the function block.
CV	This is a signed word value representing the CV output of the function block.
PV	This is a signed word value representing the process variable input to the function block.

Table 4-4. PID Function Block Data (cont'd)

Data Item	Description
Output	This is a signed word value representing the output of the function block before the application of the optional inversion. If no output inversion is configured and the output polarity bit in the control word is set to 0, this value will equal the CV output. If inversion is selected and the output polarity bit is set to 1, this value will equal the negative of the CV output.
Diff Term Storage	Used internally for storage of intermediate values. **Do not write to this location**.
Int Term Storage	Used internally for storage of intermediate values. **Do not write to this location**.
Slew Term Storage	Used internally for storage of intermediate values. **Do not write to this location**.
Clock	Internal elapsed time storage (time last executed). **Do not write to these locations**.
Lower Range	Lower range for SP, PV for faceplate display.
Upper Range	Upper range for SP, PV for faceplate display.
Reserved	Reserved for GE Fanuc use. Cannot be used for other purposes.

Initialization Values

The following table lists typical initialization values for the PID function block.

Register	Purpose	FB Units	Suggested Default	Range
%Ref+0	Loop Number		1	
%Ref+2	Sample Period	10msec	100 msec (10)	0 to 10.9 min
%Ref+3	Dead Band Selection +	Counts	320	0 to 100% of error
%Ref+4	Dead Band Selection –	Counts	320	0 to –100% of error
%Ref+5	Proportional Gain	0.01 %/%	User Tuned	0 to 327.67 %/%
%Ref+6	Derivative	0.01 seconds	User Tuned	0 to 327.67 sec
%Ref+7	Integral Rate	Repeats per 1000 sec	User Tuned	0 to 32.767 repeats/sec
%Ref+8	Bias	Counts	50% (16,000)	–100% to +100%
%Ref+9	Upper Output Clamp	Counts	100% (32,000)	–100% to +100%
%Ref+10	Lower Output Clamp	Counts	0% (0)	–100% to +100%
%Ref+11	Minimum Slew Time	Seconds per full travel	0	0 to 32,767

Description of Operation

When the PID function block is enabled, the configured execution interval (%Ref+2) is compared to the time since the last execution of the function block. If enough time has elapsed, the function block is executed. The PID loop equation is solved, based upon the actual elapsed time since the last complete execution rather than the programmed execution interval.

If the calculated control variable is beyond a configured clamp limit (%Ref+9 or %Ref+10) or has changed at a rate greater than the slew rate limit (%Ref+11), the control variable is held to the appropriate limit and the integral storage is adjusted accordingly. This is referred to as anti-reset windup.

After the control variable is calculated, it is placed in the manual register (%Ref+13) and in the control variable storage register (%Ref+16) when the control is in **AUTO** mode. When the function block is placed in **MANUAL** mode (power flow is passed to the manual input), the control variable output is held to the value in the manual register; and the manual register can be incremented or decremented by the up or down inputs to the function block. The manual register can also be loaded under program control in **MANUAL** mode.

Bumpless operation is provided between **MANUAL** and **AUTOMATIC** modes because the integral storage term is adjusted while in **MANUAL** mode, much as it is when a clamp or limit is reached. In **MANUAL** mode, the control variable output is still restricted by the configured clamps and the slew rate limit. The slew rate limit can be used to prevent an operator from trying to adjust the control variable too quickly while in **MANUAL** mode.

Difference between the PIDISA and PIDIND Blocks

The standard ISA PID algorithm (PIDISA) applies the proportional gain to each of the proportional, differential, and integral terms, as shown in the block diagram below.

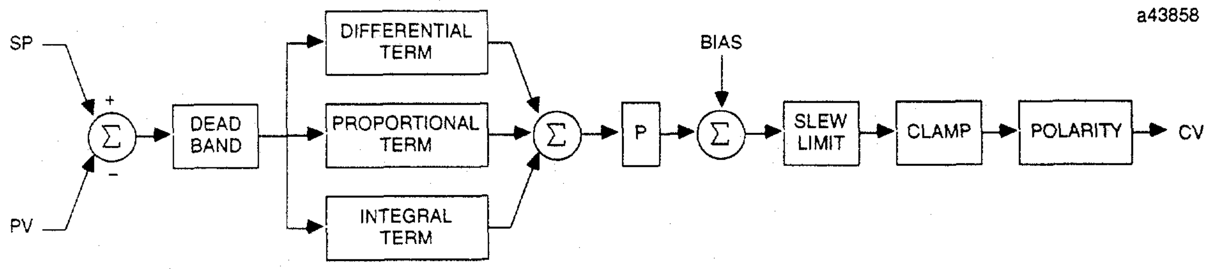


Figure 4-1. Standard ISA PID Algorithm (PIDISA)

The independent term algorithm (PIDIND) applies the proportional gain only to the proportional gain term, as shown in the block diagram below. Otherwise, the algorithms are identical.

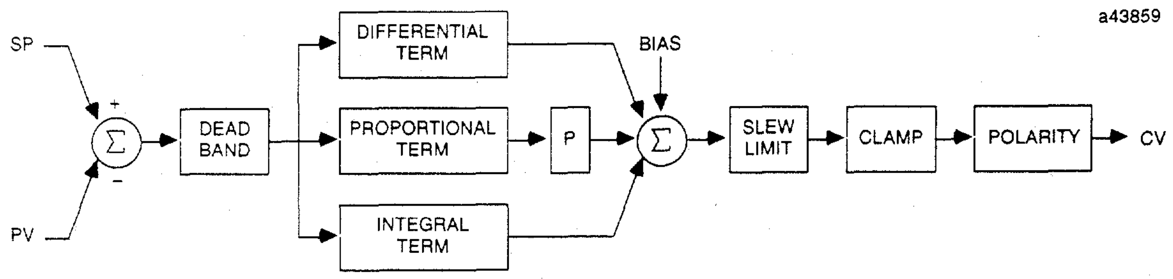
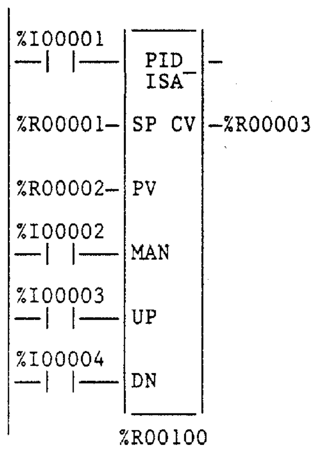


Figure 4-2. Independent Term Algorithm (PIDIND)

Example:

In the following example, the PID function is used in a rung. %R00001 contains the set point, and %R00002 contains the process variable. %R00100 is the first register in the function block. Whenever %I00001 is ON and %I00002 is OFF, the ISA PID algorithm is applied to the function's inputs and the result is placed in %R00003. Whenever %I00001 and %I00002 are ON, the result placed in CV is adjusted by the states of %I00003 and %I00004.



Note

Keep SP, CV, and PV references values outside the range of the PID lower reference array (%R00100–%R00139 in the above example).

Ziegler and Nichols Tuning Approach

Changes to the proportional gain and the integral gain will affect the output immediately. They should be adjusted slowly and in small increments to allow the system to respond to their adjustments. Loop tuning should be done according to any established method used for process control loop tuning. One such method explained below is the Ziegler and Nichols Tuning Approach.

1. Determine the process gain; apply a unit step to the control variable output and measure the process variable response after it has stabilized. This response is K, the process gain.
2. Determine the process lag time. The process lag time t can be estimated as the time it takes the process variable to begin to react to a step change in the control variable. It is typically the point at which the process variable has reached its maximum rate of change.
3. Determine the equivalent system time constant. The equivalent system time constant T can be determined by the time it takes the process variable to reach 63% of its steady state value, from a step applied to the control variable minus the process lag time t.
4. Calculate the reaction rate R:

$$R = \frac{K}{T}$$

5. For proportional control only, calculate the Proportional Gain P:

$$P = \frac{1}{(R * T)}$$

6. For proportional and integral control, calculate Proportional Gain P and Integral Gain I:

$$P = \frac{0.9}{(R * T)}$$

$$I = \frac{0.3 * P}{t}$$

These should only be used as starting values for the tuning process. These values may vary with operating points in the process, if the process is time variant or non-linear. To assure that the tuning parameters are valid, all final adjustments should be made manually and the process monitored over all operating conditions and points.

The following example illustrates how to initialize and program the PID function block.

```
|[      START OF PROGRAM LOGIC      ]
|
| If the PID function block is located at %R1, the following logic
| will initialize the data structure required by the function block.
| These values represent initial values and may be inappropriate
| for the particular process under control.
|
| P_Gain is the required Proportional Gain
| D_Gain is the required Differential Gain (Usually 0)
| I_Gain is the required Integral Gain
|
|FST_SCN  _______          _______                    _______
|--| |---|BLK_   |--------|MOVE_  |------------------|MOVE_  |-
|        | CLR_  |        | INT_  |                  | INT_  |
|        | WORD  |        |       |                  |       |
|%R00001-|IN     | CONST -|IN    Q|-%R00001   CONST -|IN    Q|-%R00003
|        | LEN   | +00001 | LEN   |           +00010 | LEN   |
|        |00035  |        |00001  |                  |00001  |
|        |_______|        |_______|                  |_______|
|
|FST_SCN  _______                   _______                    _______
|--| |---|MOVE_  |-----------------|MOVE_  |------------------|MOVE_  |-
|        | INT_  |                 | INT_  |                  | INT_  |
|        |       |                 |       |                  |       |
|P_GAIN -|IN    Q|-%R00006 D_GAIN -|IN    Q|-%R00007 I_GAIN -|IN    Q|-%R00008
|        | LEN   |                 | LEN   |                  | LEN   |
|        |00001  |                 |00001  |                  |00001  |
|        |_______|                 |_______|                  |_______|
|
|FST_SCN  _______                   _______
|--| |---|MOVE_  |-----------------|MOVE_  |-
|        | INT_  |                 | INT_  |
|        |       |                 |       |
| CONST -|IN    Q|-%R00009  CONST -|IN    Q|-%R00010
| +16000 | LEN   |          +32000 | LEN   |
|        |00001  |                 |00001  |
|        |_______|                 |_______|
|
| The PID function block can be called by just inserting it into a rung
| as shown. It will run the actual loop calculations every 100 msec
| as initialized above or every PLC sweep, whichever is longer.
|
|         _______
|--------| PID_  |-
|        | ISA_  |
|        |       |
|SET_PNT-|SP   CV|-CNT_VAR
|        |       |
|PRO_VAR-|PV     |
|        |       |
|       -|MAN    |
|        |       |
|       -|UP     |
|        |       |
|       -|DN     |
|        |_______|
|        %R00001
```

Appendix A CPU Performance Data

This appendix contains instruction and overhead timing for each Series 90-70 CPU module. This timing information can be used to predict CPU sweep times.

Instruction Timing

The Series 90-70 PLC supports many different functions and function blocks. Table A-1, beginning on the next page, lists the execution time in microseconds and the memory size in bytes for each function supported by these CPUs:

1. Model 924/925.
2. Model 914/915.
3. Models 788 and 789.
4. Models 781 and 782.
5. Models 732, 772, 731R and later, and 771P and later.

Two execution times are shown for each function:

Execution Time	Description
Enabled	Time required to execute the function or function block when power flows into and out of the function. Typically, best-case times are when the data used by the block is contained in user RAM (word-oriented memory).
Disabled	Time required to execute the function when it is not enabled.

Note

Timers are updated each time they are encountered in the logic by the amount of time consumed by the last sweep.

Table A-1. Instruction Timing

Function Group	Function	Enabled				Disabled				Increment				Size
		924/ 925	914/ 915	781/782 788/789	731/732 771/772	924/ 925	914/ 915	781/782 788/789	731/732 771/772	924/ 925	914/ 915	781/782 788/789	731/732 771/772	
Timers	ONDTR	7	10	37	79	6	8	28	55	–	–	–	–	18
	OFDT	7	10	35	69	7	10	34	66	–	–	–	–	15
	TMR	9	12	41	80	6	10	35	64	–	–	–	–	15
Counters	UPCTR	6	10	32	65	6	9	32	63	–	–	–	–	18
	DNCTR	7	10	30	65	7	10	31	62	–	–	–	–	18
Math	ADD (INT)	4	6	24	34	3	4	18	18	–	–	–	–	15
	ADD (DINT)	6	10	37	63	4	7	22	31	–	–	–	–	15
	SUB (INT)	4	6	22	35	3	4	15	19	–	–	–	–	15
	SUB (DINT)	6	10	37	63	6	8	24	31	–	–	–	–	15
	MUL (INT)	4	7	22	38	4	6	15	19	–	–	–	–	15
	MUL (DINT)	7	11	40	76	5	8	25	31	–	–	–	–	15
	DIV (INT)	5	8	24	41	4	5	16	19	–	–	–	–	15
	DIV (DINT)	7	12	42	82	5	7	23	31	–	–	–	–	15
	MOD (INT)	5	8	26	41	3	4	15	18	–	–	–	–	15
	MOD (DINT)	8	13	46	85	5	7	22	31	–	–	–	–	15
	ABS (INT)	5	8	30	50	4	6	19	26	–	–	–	–	12
	ABS (DINT)	6	9	31	52	4	6	20	26	–	–	–	–	12
	SQRT (INT)	8	14	45	82	3	5	14	16	–	–	–	–	12
	SQRT (DINT)	14	26	83	166	4	7	20	27	–	–	–	–	12
Relational	EQ (INT)	5	8	23	37	3	4	15	18	–	–	–	–	15
	EQ (DINT)	7	10	34	59	4	6	21	30	–	–	–	–	15
	NE (INT)	6	8	24	36	3	4	15	18	–	–	–	–	15
	NE (DINT)	6	10	34	58	6	7	24	30	–	–	–	–	15
	GT (INT)	7	9	27	44	3	5	14	17	–	–	–	–	15
	GT (DINT)	8	11	34	60	5	7	22	30	–	–	–	–	15
	GE (INT)	7	9	24	41	4	5	13	17	–	–	–	–	15
	GE (DINT)	6	9	31	58	5	6	20	30	–	–	–	–	15
	LT (INT)	6	9	26	44	4	4	13	17	–	–	–	–	15
	LT (DINT)	7	10	32	60	5	6	20	30	–	–	–	–	15
	LE (INT)	6	8	25	41	3	4	13	17	–	–	–	–	15
	LE (DINT)	7	10	31	58	5	7	20	30	–	–	–	–	15
	CMP (INT)	10	12	35	62	9	12	35	55	–	–	–	–	21
	CMP (DINT)	11	14	42	79	10	12	42	56	–	–	–	–	21
	RANGE (INT)	–	–	–	–	–	–	–	–	–	–	–	–	
	RANGE (DINT)	–	–	–	–	–	–	–	–	–	–	–	–	
Bit Operation	AND (WORD)	8	14	61	116	5	7	26	35	0.0	1.0	8.0	18.0	18
	AND(DWORD)	9	15	62	123	5	7	27	36	1.0	2.0	10.0	24.0	18
	OR (WORD)	8	16	62	118	6	7	26	35	0.0	1.0	9.0	19.0	18
	OR (DWORD)	10	15	62	127	6	9	26	36	1.0	2.0	11.0	27.0	18
	XOR (WORD)	7	15	58	118	5	7	23	35	0.0	1.0	9.0	19.0	18
	XOR (DWORD)	9	16	63	125	5	8	25	35	1.0	2.0	11.0	26.0	18
	NOT (WORD)	5	10	40	76	5	6	23	32	0.0	0.0	3.0	8.0	15
	NOT (DWORD)	8	12	40	79	6	7	22	31	0.0	0.0	3.0	10.0	15
	MCMP (WORD)	12	23	98	212	7	11	36	58	0.0	1.0	3.0	8.0	30
	MCMP (DWORD)	16	25	100	220	8	11	36	56	1.0	2.0	7.0	16.0	30
	SHL (WORD)	11	17	63	137	5	8	30	41	0.0	1.0	4.0	10.0	24
	SHL (DWORD)	13	20	70	146	6	10	31	43	1.0	2.0	8.0	20.0	24
	SHR (WORD)	12	19	64	138	5	9	27	42	0.0	1.0	4.0	10.0	24
	SHR (DWORD)	13	20	67	147	7	9	28	43	1.0	2.0	8.0	20.0	24
	ROL (WORD)	12	16	57	119	5	6	24	53	0.0	1.0	4.0	9.0	18
	ROL (DWORD)	14	19	63	127	5	8	24	35	1.0	2.0	8.0	19.0	18
	ROR (WORD)	9	15	53	113	5	6	23	34	0.0	1.0	4.0	9.0	18
	ROR (DWORD)	12	16	59	122	5	8	24	36	1.0	2.0	8.0	19.0	18
	BTST (WORD)	8	13	46	91	5	8	26	36	–	–	–	–	18
	BTST (DWORD)	8	13	45	90	6	9	23	34	–	–	–	–	18
	BSET (WORD)	7	12	45	89	6	9	24	34	–	–	–	–	15
	BSET (DWORD)	8	13	45	89	5	8	22	31	–	–	–	–	15
	BCLR (WORD)	7	14	44	89	2	7	24	33	–	–	–	–	15
	BCLR (DWORD)	8	12	43	89	5	8	22	32	–	–	–	–	15
	MOVE (BIT)	13	19	58	119	5	7	20	32	0.0	1.0	3.0	6.0	15
	BPOS (WORD)	9	15	55	107	5	7	25	35	0.0	0.0	1.0	4.0	18
	BPOS (DWORD)	13	20	81	158	5	8	23	36	0.0	0.0	3.0	6.0	18
	SHFR (BIT)	11	18	74	162	4	5	15	28	0.3	0.7	3.1	6.1	24

Note:
1. Time (in microseconds) is based on Release 5.0 of Logicmaster 90-70 software. For information not available at print time for this manual (–), refer to the IPI for that CPU.
2. For table functions, increment is in units of length specified. For bit operation functions, microseconds/bit. For data move functions, microseconds/the number of bits or words.
3. Enabled time is for single length units of type %R.
4. COMMREQ time has been measured between CPU and PCM with NOWAIT option.
5. DOIO is the time to output values to discrete output module.

Table A-1. Instruction Timing (cont'd)

Function Group	Function	Enabled 924/925	Enabled 914/915	Enabled 781/782 788/789	Enabled 731/732 771/772	Disabled 924/925	Disabled 914/915	Disabled 781/782 788/789	Disabled 731/732 771/772	Increment 924/925	Increment 914/915	Increment 781/782 788/789	Increment 731/732 771/772	Size
Data Move	MOVE (WORD)	4	7	19	29	3	4	13	18	0.1	0.1	0.4	0.8	15
	MOVE (DWORD)	8	11	33	56	6	8	23	31	0.2	0.2	1.0	1.7	15
	BLKMOV (WORD)	7	8	24	36	3	4	13	16	–	–	–	–	30
	BLKMOV (DWORD)	8	12	35	55	7	10	33	50	–	–	–	–	44
	SWAP (WORD)	8	11	28	45	3	5	14	17	0.3	0.7	1.7	3.5	15
	SWAP (DWORD)	8	12	36	65	5	7	20	31	0.5	0.9	2.7	5.6	15
	BLKCLR	5	7	19	30	3	4	12	16	0.1	0.2	0.6	1.4	12
	BITSEQ	12	16	47	96	7	11	40	80	–	–	–	–	24
	SHFR (WORD)	10	16	55	114	5	6	16	28	0.0	0.0	0.3	0.6	24
	SHFR (DWORD)	11	17	56	117	5	6	18	30	0.1	0.2	0.8	1.3	24
	SORT	258	420	1406	2896	5	7	22	31	2.0	3.7	14.5	36.9	15
Data Table	TBLRD (INT)	5	8	32	53	3	4	12	17	–	–	–	–	21
	TBLRD (DINT)	8	11	31	58	4	6	13	18	–	–	–	–	21
	TBLWR (INT)	7	11	35	60	4	6	11	17	–	–	–	–	21
	TBLWR (DINT)	9	15	54	105	5	8	26	37	–	–	–	–	21
	FIFORD (INT)	6	9	30	58	2	4	10	17	–	–	–	–	21
	FIFORD (DINT)	6	10	31	61	4	5	10	18	–	–	–	–	21
	FIFOWRT (INT)	5	9	28	55	2	4	10	17	–	–	–	–	21
	FIFOWRT (DINT	10	14	43	85	5	8	22	37	–	–	–	–	21
	LIFORD (INT)	5	8	26	52	2	4	10	17	–	–	–	–	21
	LIFORD (DINT)	6	9	30	57	3	5	15	18	–	–	–	–	21
	LIFOWRT (INT)	5	9	27	55	3	4	13	17	–	–	–	–	21
	LIFOWRT (DINT)	8	13	47	85	6	10	28	37	–	–	–	–	21
Array	ARRAT_MOVE (BIT)	12	22	91	197	6	9	30	41	0.0	0.0	0.3	0.7	–
	ARRAY_MOVE (BYTE)	12	20	76	163	6	9	28	43	0.2	0.4	1.2	2.4	–
	ARRAY_MOVE (WORD)	12	21	78	164	5	9	27	42	0.0	0.1	0.3	0.6	–
	ARRAY_MOVE (DWORD)	13	22	77	165	6	9	29	43	0.1	0.1	0.7	1.3	–
	SRCH (BYTE)	9	16	59	119	5	8	26	41	0.0	0.1	0.6	1.2	–
	SRCH (WORD)	10	16	59	122	5	9	27	41	0.0	0.1	0.5	1.2	–
	SRCH (DWORD)	10	16	62	132	6	9	26	41	0.2	0.5	1.7	3.0	–
	ARRAY_RANGE (WORD)	–	–	–	–	–	–	–	–	–	–	–	–	–
	ARRAY_RANGE (DWORD)	–	–	–	–	–	–	–	–	–	–	–	–	–
Conversion	to INT (UINT)	4	7	23	32	3	4	15	16	–	–	–	–	12
	to INT (DINT)	5	8	25	40	4	6	20	26	–	–	–	–	12
	to INT (BCD-4)	6	9	27	38	4	5	14	16	–	–	–	–	12
	to DINT (INT)	5	8	24	33	4	4	15	17	–	–	–	–	12
	to DINT (UINT)	6	9	23	33	3	4	13	17	–	–	–	–	12
	to DINT (BCD-8)	8	13	42	76	5	7	18	26	–	–	–	–	12
	to UINT (INT)	5	8	22	32	3	5	14	17	–	–	–	–	12
	to UINT (DINT)	6	9	24	40	4	6	20	26	–	–	–	–	12
	to UINT (BCD-4)	6	9	24	38	3	5	12	16	–	–	–	–	12
	to BCD-4 (INT)	6	9	27	44	3	5	12	16	–	–	–	–	12
	to BCD-4 (UINT)	5	8	24	43	3	4	12	16	–	–	–	–	12
	to BCD-8 (DINT)	10	15	39	77	5	8	18	26	–	–	–	–	12

Note:
1. Time (in microseconds) is based on Release 5.0 of Logicmaster 90-70 software. For information not available when this manual was being printed (represented by a dash: –), refer to the IPI for each CPU.
2. *For table functions, increment is in units of length specified. For bit operation functions, microseconds/bit. For data move functions, microseconds/the number of bits or words.*
3. Enabled time is for single length units of type %R.
4. COMMREQ time has been measured between CPU and PCM with NOWAIT option.
5. DOIO is the time to output values to discrete output module.

Table A-1. Instruction Timing (cont'd)

Function Group	Function	Enabled				Disabled				Increment				Size
		924/ 925	914/ 915	781/782 788/789	731/732 771/772	924/ 925	914/ 915	781/782 788/789	731/732 771/772	924/ 925	914/ 915	781/782 788/789	731/732 771/772	
Control	JUMP	3	3	7	13	3	3	7	6	–	–	–	–	–
	FOR/NEXT	13	21	63	124	8	9	29	53	–	–	–	–	–
	MCR/ENDMCR Combined	6	9	31	51	7	10	32	47	–	–	–	–	9
	DOIO	34	50	141	304	4	6	17	20	–	–	–	–	15
	DOIO with ALT	35	53	136	295	3	6	14	20	–	–	–	–	15
	SUSIO	5	7	14	22	4	3	12	14	–	–	–	–	6
	COMMREQ	140	190	370	810	9	11	25	38	–	–	–	–	18
	CALL/RETURN (LD)	26	34	92	229	2	4	6	6	–	–	–	–	15
	CALL/RETURN (SFC)	92	132	430	763	3	3	5	7	–	–	–	–	15
	CALL/RETURN (PSB)	17	20	61	133	3	3	4	9	–	–	–	–	15
	CALL/RETURN (External Block)	51	89	300	337	3	3	4	8	–	–	–	–	15
	PIDISA	63	119	463	966	7	10	39	65	–	–	–	–	27
	PIDIND	68	130	505	1046	7	12	43	67	–	–	–	–	27
	VMERD (BYTE)	17	28	92	173	6	8	23	35	0.8	0.8	0.9	1.2	18
	VMERD (WORD)	17	28	94	171	6	8	23	34	0.8	0.8	0.9	1.2	18
	VMEWRT (BYTE)	19	32	102	192	5	8	24	37	0.8	0.8	0.7	0.9	18
	VMEWRT (WORD)	17	31	98	188	4	8	22	34	0.8	0.8	0.8	0.9	18
	VMERMW	19	32	94	192	4	7	21	35	–	–	–	–	18
	VMETST	21	30	82	156	6	8	24	39	–	–	–	–	15
	VME_CFG_RD	32	51	166	344	6	9	27	45	–	–	–	–	–
	VME_CFG_WRT	26	43	154	333	6	9	27	45	–	–	–	–	–
	SVCREQ:									–	–	–	–	12
	#1	14	26	81	96	4	5	17	27	–	–	–	–	12
	#2	13	25	76	86	5	7	17	26	–	–	–	–	12
	#3	13	25	75	85	4	7	17	25	–	–	–	–	12
	#4	13	25	76	85	4	7	18	27	–	–	–	–	12
	#6	15	27	82	92	4	6	17	27	–	–	–	–	12
	#7	35	57	157	304	5	6	16	25	–	–	–	–	12
	#8	30	49	145	280	4	6	16	26	–	–	–	–	12
	#9	27	43	119	184	5	7	16	26	–	–	–	–	12
	#10	16	30	89	125	4	7	18	25	–	–	–	–	12
	#11	14	27	80	115	4	6	18	25	–	–	–	–	12
	#12	14	25	76	83	4	5	17	26	–	–	–	–	12
	#13	–	–	–	–	–	–	–	–	–	–	–	–	12
	#14	173	311	913	2561	5	6	15	25	–	–	–	–	12
	#15	28	32	99	293	5	6	17	27	–	–	–	–	12
	#16	23	37	105	130	5	5	19	27	–	–	–	–	12
	#17	24	42	134	43	4	6	17	25	–	–	–	–	12
	#18	2941	3002	3212	375	5	6	17	24	–	–	–	–	12
	#19	14	25	77	92	4	6	18	25	–	–	–	–	12
	#20	23	41	125	253	4	6	15	25	–	–	–	–	12
	#21	173	243	685	1554	4	6	17	25	–	–	–	–	12
	#22	14	24	73	83	4	6	18	25	–	–	–	–	12

Note:
1. Time (in microseconds) is based on Release 5.0 of Logicmaster 90-70 software. For information not available when this manual was being printed (represented by a dash: –), refer to the IPI for each CPU.
2. For table functions, increment is in units of length specified. For bit operation functions, microseconds/bit. For data move functions, microseconds/the number of bits or words.
3. Enabled time is for single length units of type %R.
4. COMMREQ time has been measured between CPU and PCM with NOWAIT option.
5. DOIO is the time to output values to discrete output module.

Table A-1. Instruction Timing (cont'd)

Function Group	Function	Enabled				Disabled				Increment				
		924/ 925	914 915	781/782 788/789	731/732 771/772	924/ 925	914/ 915	781/782 788/789	731/732 771/772	924/ 925	914/ 915	781/782 788/789	731/732 771/772	Size
Floating Point	Math:													
	ADD_REAL	8	13	49	108	5	7	21	31	–	–	–	–	–
	SUB_REAL	8	12	49	108	5	7	22	31	–	–	–	–	–
	MUL_REAL	8	12	50	108	5	7	22	31	–	–	–	–	–
	DIV_REAL	8	13	53	109	5	7	25	31	–	–	–	–	–
	ABS_REAL	7	10	44	85	5	6	20	26	–	–	–	–	–
	SQRT_REAL	8	12	46	91	5	7	21	27	–	–	–	–	–
	Trigonometric:													
	SIN	11	21	71	126	4	6	19	25	–	–	–	–	–
	COS	11	21	63	121	4	6	19	25	–	–	–	–	–
	TAN	10	19	58	115	4	7	19	25	–	–	–	–	–
	ASIN	9	15	54	132	5	7	19	25	–	–	–	–	–
	ACOS	10	18	67	164	4	7	18	25	–	–	–	–	–
	ATAN	12	21	55	111	4	6	20	25	–	–	–	–	–
	Logarithmic:													
	LOG	8	13	45	101	4	6	16	27	–	–	–	–	–
	LN	8	13	47	99	4	7	16	25	–	–	–	–	–
	EXPT	12	22	189	228	4	6	19	30	–	–	–	–	–
	EXP	12	23	86	194	4	6	16	25	–	–	–	–	–
	Comparison:													
	EQ_REAL	9	12	41	87	5	7	20	31	–	–	–	–	–
	NE_REAL	8	11	40	85	5	7	20	31	–	–	–	–	–
	GT_REAL	9	12	42	92	6	8	22	31	–	–	–	–	–
	GE_REAL	8	11	44	92	6	7	24	31	–	–	–	–	–
	LT_REAL	9	12	43	90	6	7	23	31	–	–	–	–	–
	LE_REAL	8	11	43	90	6	8	21	31	–	–	–	–	–
	CMP_REAL	18	21	58	114	15	17	41	64	–	–	–	–	–
	Data Move:													
	MOVE_REAL	6	11	35	56	6	8	21	30	–	–	–	–	–
	Conversion:													
	REAL_TO_INT	7	12	45	96	5	6	18	26	–	–	–	–	–
	REAL_TO_UINT	7	13	50	112	4	7	19	25	–	–	–	–	–
	REAL_TO_DINT	8	14	42	94	5	8	18	26	–	–	–	–	–
	INT_TO_REAL	8	12	35	68	5	8	18	26	–	–	–	–	–
	UINT_TO_REAL	6	12	36	70	4	7	17	26	–	–	–	–	–
	DINT_TO_REAL	6	10	35	65	4	6	18	25	–	–	–	–	–
	REAL_TRUN_INT	9	14	55	126	5	6	17	26	–	–	–	–	–
	REAL_TRUN_DINT	8	13	53	121	5	7	17	26	–	–	–	–	–
	DEG_TO_RAD	7	12	45	101	4	7	17	26	–	–	–	–	–
	RAD_TO_DEG	8	13	47	105	4	6	17	26	–	–	–	–	–
	BCD4_TO_REAL	7	13	39	92	4	6	15	26	–	–	–	–	–
	BCD8_TO_REAL	8	15	43	100	4	8	18	26	–	–	–	–	–

Note:
1. Time (in microseconds) is based on Release 5.0 of Logicmaster 90-70 software. For information not available when this manual was being printed (represented by a dash: –), refer to the IPI for each CPU.
2. For table functions, increment is in units of length specified. For bit operation functions, microseconds/bit. For data move functions, microseconds/the number of bits or words.
3. Enabled time is for single length units of type %R.
4. COMMREQ time has been measured between CPU and PCM with NOWAIT option.
5. DOIO is the time to output values to discrete output module.

Overhead Sweep Impact Time

This part of the appendix contains overhead timing information for the Series 90-70 PLC CPU. This information can be used in conjunction with the estimated logic execution time to predict sweep times for each of the Series 90-70 CPUs. The information in this section is made up of a base sweep time plus sweep impact times for each of the CPU models: 731, 732, 771, 772, 781, and 782. The predicted sweep time is computed by adding the sweep impact time(s), the base sweep, and the estimated logic execution time.

$\boxed{\texttt{Predicted Sweep}} = \boxed{\texttt{Base Sweep}} + \boxed{\texttt{Sweep Impact Times}} + \boxed{\texttt{Estimated Logic Execution Time}}$

Two examples of predicting sweep times are provided at the end of this appendix.

Sweep impact times are composed of four basic sections:

1. Programmer communications sweep impact.
2. I/O scan and fault sweep impact.
3. Intelligent Option Module (PCMs and LAN modules) sweep impact.
4. I/O interrupt performance and sweep impact.

Each of these sections describes the functions and provides tables with the corresponding times for each CPU model. Note that the tables have only two columns: 731/732/771/772 and 781/782. Earlier versions of 731 and 771 hardware are slower.

The information in these tables may be used to predict sweep time based on a given configuration.

What the Tables Contain

The following tables contain sweep impact times for overhead functions for the Series 90-70 PLC. Base sweep time is the time for an empty _MAIN program block to execute, with no configuration stored and none of the windows active. The rest of the timing values are given as sweep impact times, that is, the time added to the sweep by the function in question. Sweep impact times are nominal.

Note

There are two categories of sweep impact numbers listed in the tables, those which impact the sweep every sweep and those which impact the sweep only when invoked. The functions which impact the sweep every sweep are listed in bold type in each table.

In some of the tables, functions are shown as asynchronously impacting the sweep. This means that there is not a set phase of the sweep in which the function takes place. For instance, the scanning of all I/O modules takes place during either the input or output scan phase of the PLC CPUs sweep. However, I/O interrupts are totally asynchronous to the sweep and will interrupt any function currently in progress.

The communication functions (with the exception of the high priority programmer requests) are all processed within one of the two windows in the sweep (the programmer communications window and the system communications window). Sweep impact times for the various service requests are all minimum sweep impact times for the defined functions, where the window times have been adjusted so that no timeslicing (limiting) of the window occurs in a given PLC sweep. This means that, as much as possible, each function is completed in one occurrence of the window (between consecutive logic scans). The sweep impact of these functions can be spread out over multiple sweeps (limited) by adjusting the window times to a value lower than the documented sweep impact time. For the programmer, the default time is 10 ms; therefore, some of the functions listed in that section will naturally timeslice over successive sweeps.

Base Sweep Times

The base sweep time for each CPU model is shown below. This time is for an empty _MAIN program block with no programmer attached, no configuration downloaded, and no other module present in the system other than the CPU. The following diagram shows the full sweep phases and the base sweep phases contrasted so that the optional parts of the sweep are illustrated.

Table A-2. Base Sweep vs. Full Sweep Phases

BASE SWEEP	FULL SWEEP
<START OF SWEEP>	<START OF SWEEP>
Sweep Housekeeping	Sweep Housekeeping
↓	↓
NULL Input Scan *	Input Scan *
↓	↓
Program Logic Execution	Program Logic Execution
↓	↓
NULL Output Scan *	Output Scan *
↓	↓
↓	Poll for Missing I/O Modules **
↓	↓
↓	Programmer Communications Window
↓	↓
↓	System Communications Window
<END OF SWEEP>	<END OF SWEEP>

* If I/O is suspended, then the input and output scans are skipped.
** Polling for missing I/O modules only occurs if a "Loss of ..." fault has been logged for a Series 90-70 I/O module.

For the base sweep, the lack of configuration means that the input and output scan phases of the sweep are NULL (i.e. check for configuration and then end). The presence of a configuration with no I/O modules or intelligent I/O modules (GBC, PSM, etc.) would have the same effect. The logic execution time is not zero in the base sweep. The time to execute the empty _MAIN program is included so that you only need to add the estimated execution times of the functions actually programmed. The base sweep also assumes no missing I/O modules. The lack of programmer attachment means that the programmer communications window is never opened. The lack of intelligent option modules means that the system communications window is never opened.

The following table gives the base sweep times in milliseconds for each CPU model.

Table A-3. Base Sweep Times

CPU Model	924/ 925	914/ 915	781/782 788/789	731/732 771/772
Base Sweep Time				2.00

Note

Not all of the sweep time information was available at the time this manual was printed (the blank spaces in the tables above and below). Refer to the IPI for the specific CPU for this information.

Programmer Sweep Impact Times

The following table shows the programmer sweep impact times in milliseconds. The times are broken up into parallel and serial since these interfaces are significantly different. (Each item in the table is described in more detail at the end of the table.)

Table A-4. Programmer Sweep Impact Times

Sweep Impact Item	924/ 925	914/ 925	781/782 788/789	731/731 771/772
Parallel Programmer				
Programmer window				0.75
Reference table monitor				4.90
Editor monitor				4.90
Word-for-word change				28.10
ALT-S store				14.80
High priority request				9.90
Serial programmer				
Programmer Window				0.50
Reference table monitor				5.60
Editor monitor				6.40

Note

Functions in bold type in Table A-4 above impact the sweep continuously. All other functions impact the sweep only when invoked.

Each of the items included in the table is described below.

Sweep Impact Item	Description
Programmer Window	**Parallel:** The time to open the programmer window but not process any requests. The programmer is attached with a parallel connection; no reference values are being monitored. **Serial:** Same as above except attached serially. Unlike the parallel programmer, the sweep impact cannot be limited to a single sweep. Serial communications, in addition to the service processing time that is performed inside the programmer communications window, has processing time associated with the receiving of data over the RS422/485 serial link. This serial communications processing time is asynchronous to the sweep in order to meet the timing requirements of the serial protocol.
Reference Table Monitor	The sweep impact to refresh the reference table screen. (The %R table was used as the example.) Mixed table display impacts are slightly larger. The sweep impact may not be continuous, depending on the sweep time of the PLC and the speed of the Logicmaster 90 host.
Editor Monitor	The sweep impact to refresh the editor screen when monitoring ladder logic. The times given in the table are for a logic screen containing one contact, two coils, and eleven registers. As with the reference table sweep impact, the impact may not be continuous.
Word-for-Word Change	The sweep impact to change a constant input on a MOVE_UINT function from 1 to 2. This is the smallest change that can be made. A change to a coil requires updates to the coil use and retentive maps contained in the PLC. If the %Q or %M reference address is changed on a function block, then the size of the change to the coil use and retentive maps can be quite large. A large word-for-word change will have very little sweep impact if done with the parallel programmer (worst-case of 35 ms on a 731 CPU). A large word-for-word change will have a big sweep impact if done with the serial programmer. The time is linear for the number of bytes in the request with the worst case being slightly larger than the worst case for parallel.
ALT-S Store	The sweep impact to store a 933 byte program block. The sweep impact is linear as the program block gets larger.
High Priority Request (Parallel only)	There are several programmer high priority requests: change PLC mode (includes ALT-R), read constant sweep state/time, read window times, change constant sweep state/time, and change window times. Unlike the other programmer service requests, which are serviced inside the programmer communications window, high priority requests interrupt the sweep and are serviced asynchronously at the time they are issued by the programmer. The worst case high priority request is monitoring/changing the window times. This time is shown below.

I/O Scan and I/O Fault Sweep Impact

The I/O scan sweep impact has two parts, Series 90-70 I/O and Genius I/O. The equation for computing I/O scan sweep impact is:

$$\boxed{\texttt{I/O Scan Sweep Impact}} = \boxed{\texttt{I/O Scan Overhead}} + \boxed{\texttt{Series 90-70 I/O Scan Impact}} + \boxed{\texttt{Genius I/O Scan Impact}}$$

The following table shows the I/O scan overhead in milliseconds for each CPU:

Table A-5. I/O Scan Overhead *

CPU Model	924/ 925	914/ 915	781/782 788/789	731/732 771/772
I/O scan overhead				0.38

* Times are in milliseconds. For information not available when this manual was being printed (represented by a dash: –), refer to the IPI for each CPU.

Note

I/O scan overhead impacts the sweep continuously.

Sweep Impact of Series 90-70 I/O Modules

The I/O scan of the Series 90-70 I/O modules is impacted as much by location and reference address of a module as it is by the number of modules. The I/O scan has several basic parts.

I/O Scan	Description
I/O Scan Overhead	Includes the setup for input and output scan and the selection of the main rack.
Rack Setup Time	Each expansion rack is selected separately because of the addressing of expansion racks on the VME bus. This results in a fixed overhead per expansion rack, regardless of the number of modules in that rack.
Per Module Setup Time	Each Series 90-70 I/O module has a fixed setup scan time.
Byte Transfer Time	The actual transfer of bytes is much faster for modules located in the main rack than for those in expansion racks. The byte transfer time differences will be accounted for by using different times for I/O modules in the main rack versus expansion racks.

In addition, analog input expander modules (the same as Genius blocks) have the ability to be grouped into a single transfer as long as consecutive reference addresses are used for modules that have consecutive slot addresses. Each sequence of consecutively addressed modules is called a scan segment. There is a time penalty for each additional scan segment.

This leads to the following form, which can be used for computing I/O module sweep impact. The calculation contains times for analog input expanders that are either grouped into the same scan segment as the preceding module or are grouped in a separate new scan segment. The sweep impact times can be found in table A-7.

Table A-6. Worksheet A: I/O Module Sweep Time

Number of expansion racks		______		
Sweep impact per expansion rack	x	______	=	______
Number of discrete I/O modules — main rack		______		
Sweep impact per discrete I/O module — main rack	x	______	=	______
Number of discrete I/O modules — expansion rack		______		
Sweep impact per discrete I/O module — expansion rack	x	______	=	______
Number of analog input base and output modules — main rack		______		
Sweep impact per analog input base and output module — main rack	x	______	=	______
Number of analog input expander modules (same segment) — main rack		______		
Sweep impact per analog input expander modules (same segment) — main rack	x	______	=	______
Number of analog input expander modules (new segment) — main rack		______		
Sweep impact per analog input expander modules (new segment) — main rack	x	______	=	______
Number of analog input base and output modules — expansion rack		______		
Sweep impact per analog input base and output module — expansion rack	x	______	=	______
Number of analog input base and output modules (same segment) — expansion rack		______		
Sweep impact per analog input base and output module (same segment) — expansion rack	x	______	=	______
Number of analog input base and output modules (new segment) — expansion rack		______		
Sweep impact per analog input base and output module (new segment) — expansion rack	x	______	=	______
Predicted Series 90-70 I/O Module Sweep Impact				______

Note

If point faults are enabled, substitute the corresponding times for point faults enabled, as shown in the following table.

An approximate per point or per channel average is shown in the following tables. These averages are based on 1024 points (512 in and 512 out) for discrete and 128 channels (96 in and 32 out) for analog. The 96 analog input channels consist of two base modules and five expanders. Actual values will vary from the approximate average, depending on the system I/O configuration.

Note

Not all of the sweep time information was available at the time this manual was printed (the blank spaces in the table below). Refer to the IPI for the specific CPU for this information.

Table A-7. Sweep Impact Time for Model 70 I/O Modules and Racks *

	924/ 925	914/ 915	781/782 788/789	731/732 771/772
Rack Setup				
per expansion rack				0.05
Discrete I/O Modules				
per I/O module in main rack				0.08
per I/O module in main rack w/point faults enabled				0.12
per I/O module in expansion rack				0.09
per I/O module in expansion rack w/point faults enabled				0.13
per fault message **				1.70
Rough Average per Point (no point faults)				3.1µs
Rough Average per Point (w/ point faults)				4.4µs
Analog I/O Modules				
per input/output module in main rack				0.08
per input/output module w/point faults enabled in main rack				0.12
per input or output module in expansion rack				0.10
per input/output module w/point faults enabled in expansion rack				0.15
per input expander module in same segment in main rack				0.02
per input expander module w/point faults enabled in same segment in main rack				0.03
per input expander module in new segment in main rack				0.03
per input expander module w/point faults enabled in new segment in main rack				0.05
per input expander module in same segment in expansion rack				0.06
per input expander module w/point faults enabled in same segment in expansion rack				0.10
per input expander module in new segment in expansion rack				0.10
per input expander module w/point faults enabled in new segment in expansion rack				0.15
per fault message **				2.10
Rough Average per Channel (no point faults)				10.8µs
Rough Average per Channel (w/ point faults)				14.0µs

* Times are in milliseconds, except for those identified as microseconds.

** Faults for discrete Series 90-70 I/O modules are always polled for by the PLC CPU. When one occurs, it is always logged during one of the I/O scan phases of the sweep. These faults are only polled when point faults are enabled.

Note

Functions in bold type in Table A-7 above impact the sweep continuously. All other functions impact the sweep only when invoked.

Sweep Impact of Genius I/O and GBCs

The sweep impact of Genius I/O and Genius Bus Controllers (GBC) is similar to that of Series 90-70 I/O. There is an overhead for the I/O scan that should be counted only once between the Series 90-70 I/O scan and the Genius I/O scan. There is also a per Genius Bus Controller sweep impact, a per scan segment sweep impact; and a transfer time (per word) sweep impact for all I/O data.

The sweep impact per Genius Bus Controller has three parts:

1. Sweep impact to open the system communications window. This is added only once when the first intelligent option module (of which the Genius Bus Controller is one) is placed in the system.
2. Sweep impact to poll each Genius Bus Controller for background messages (datagrams). This part is an impact for every Genius Bus Controller in the system.

Note

Both the first and second parts of the Genius Bus Controller's sweep impact may be eliminated by closing the system communications window (setting its time to 0). This should only be done to reduce scan time during critical phases of a process to ensure minimal scan time. Incoming messages will timeout, and COMMREQs will stop working while the window is closed. Communications with PCM and LAN modules will also stop.

3. Sweep impact to scan the Genius Bus Controller. This impact results from the PLC CPU notifying the Genius Bus Controller that its new output data has been transferred and commanding the Genius Bus Controller to ready its input data, as well as informing the Genius Bus Controller that the PLC has finished another sweep and is still in RUN mode.

A scan segment for Genius I/O consists of Genius blocks on the same bus with consecutive reference addresses and consecutive bus addresses. The time to process a single scan segment is higher for an input scan segment than it is for an output scan segment. The scan segment processing is the same for analog, discrete, and global data scan segments. Discrete data is transferred a byte at a time and takes longer to complete the transfer than analog data, which is transferred a word at a time. Global data should be counted as either discrete or analog, based on the memory references used in the source or destination.

Note

Not all of the sweep time information was available at the time this manual was printed (the blank spaces in the table below). Refer to the IPI for the specific CPU for this information.

Table A-8. Sweep Impact Time of Genius I/O and GBCs *

	924/ 925	914/ 915	781/782 788/789	731/732 771/772
Genius Bus Controller				
open system communications window				0.60
per Genius Bus Controller polling for background messages				0.10
per Genius Bus Controller I/O Scan				1.70
First Genius Bus Controller **				2.40
Subsequent Genius Bus Controllers				1.60
Genius I/O Blocks				
per input block scan segment				0.03
per input block scan segment w/point faults enabled				0.09
per output block scan segment				0.03
per output block scan segment w/point faults enabled				0.09
per byte discrete I/O data in the main rack				3.2μs
per byte discrete I/O data in expansion racks				4.5μs
per word analog I/O data in the main rack				2.9μs
per word analog I/O data in expansion racks				7.7μs
Asynchronous Events				
Fault Message				2.00

* Times are in milliseconds, except for those identified as microseconds.
** The extra time for the first GBC is the same time as shown in the next table for the first intelligent option module. This time should be counted only once.

Note

Functions in bold type in Table A-8 above impact the sweep continuously. All other functions impact the sweep only when invoked.

Use the following form for predicting the sweep impact due to Genius I/O. The sweep impact times can be found in table A-8.

Table A-9. Worksheet B: Genius I/O Sweep Time

Open system communications window		_______	=	_______
GBC I/O scan		_______		
GBC poll for background messages	x	_______	=	_______
Number of GBCs	x	_______	=	_______
Input block scan segments - number of		_______		
Input block scan segments - sweep impact	x	_______	=	_______
Output block scan segments - number of		_______		
Output block scan segments - sweep impact	x	_______	=	_______
Bytes of discrete I/O data on GBCs - main rack		_______		
Sweep impact/byte,discrete I/O data - main rack	x	_______	=	_______
Bytes of discrete I/O data on GBCs - expansion racks		_______		
Sweep impact/bytes of discrete I/O data - expansion racks	x	_______	=	_______
Words of analog I/O data on GBCs - main rack		_______		
Sweep impact/word analog I/O data - main rack	x	_______	=	_______
Words of analog I/O data on GBCs - expansion racks		_______		
Sweep impact/word analog I/O data - expansion racks	x	_______	=	_______
Predicted Genius I/O Scan Impact				_______

Sweep Impact of FIP I/O and FBCs

The sweep impact of FIP I/O and FIP Bus Controllers (FBC) is similar to that of Series 90-70 I/O. There is an overhead for the I/O scan that should be counted only once between the Series 90-70 I/O scan and the FBC I/O scan. There is also a per FIP Bus Controller sweep impact, a per scan segment sweep impact; and a transfer time (per word) sweep impact for all I/O data.

The sweep impact per FIP Bus Controller has three parts:

1. Sweep impact to open the system communications window. This is added only once when the first intelligent option module (of which the FIP Bus Controller is one) is placed in the system.
2. Sweep impact to poll each FIP Bus Controller for background messages (datagrams). This part is an impact for every FIP Bus Controller in the system.

Note

Both the first and second parts of the FIP Bus Controller's sweep impact may be eliminated by closing the system communications window (setting its time to 0). This should only be done to reduce scan time during critical phases of a process to ensure minimal scan time. Incoming messages will timeout, and COMMREQs will stop working while the window is closed. Communications with PCM and LAN modules will also stop.

3. Sweep impact to scan the FIP Bus Controller. This impact results from the PLC CPU notifying the FIP Bus Controller that its new output data has been transferred and commanding the FIP Bus Controller to ready its input data, as well as informing the FIP Bus Controller that the PLC has finished another sweep and is still in RUN mode.

A scan segment for FIP I/O consists of FIP blocks on the same bus with consecutive reference addresses and consecutive bus addresses. The time to process a single scan segment is higher for an input scan segment than it is for an output scan segment. The scan segment processing is the same for analog, discrete, and global data scan segments. Discrete data is transferred a byte at a time and takes longer to complete the transfer than analog data, which is transferred a word at a time. Global data should be counted as either discrete or analog, based on the memory references used in the source or destination.

Note

The sweep time information was not available at the time this manual was printed (the blank spaces in the table below). Refer to the IPI for the specific CPU for this information.

Table A-10. Sweep Impact Time of FIP I/O and FBCs *

	924/ 925	914/ 915	781/782
FIP Bus Controller			
open system communications window			
per FIP Bus Controller polling for background messages			
per FIP Bus Controller I/O Scan			
First FIP Bus Controller **			
Subsequent FIP Bus Controllers			
FIP I/O Blocks			
per input block scan segment			
per input block scan segment w/point faults enabled			
per output block scan segment			
per output block scan segment w/point faults enabled			
per byte discrete I/O data in the main rack			
per byte discrete I/O data in expansion racks			
per word analog I/O data in the main rack			
per word analog I/O data in expansion racks			
Asynchronous Events			
Fault Message			

* Times are in milliseconds, except for those identified as microseconds.

** The extra time for the first FBC is the same time as shown in the next table for the first intelligent option module. This time should be counted only once.

Note

Functions in bold type in Table A-8 above impact the sweep continuously. All other functions impact the sweep only when invoked.

Use the following form for predicting the sweep impact due to FIP I/O. The sweep impact times can be found in table A-8.

Table A-11. Worksheet B: FIP I/O Sweep Time

Open system communications window		______	=	______
FBC I/O scan		______		
FBC poll for background messages	x	______	=	______
Number of FBCs	x	______	=	______
Input block scan segments - number of		______		
Input block scan segments - sweep impact	x	______	=	______
Output block scan segments - number of		______		
Output block scan segments - sweep impact	x	______	=	______
Bytes of discrete I/O data on FBCs - main rack		______		
Sweep impact/byte,discrete I/O data - main rack	x	______	=	______
Bytes of discrete I/O data on FBCs - expansion racks		______		
Sweep impact/bytes of discrete I/O data - expansion racks	x	______	=	______
Words of analog I/O data on FBCs - main rack		______		
Sweep impact/word analog I/O data - main rack	x	______	=	______
Words of analog I/O data on FBCs - expansion racks		______		
Sweep impact/word analog I/O data - expansion racks	x	______	=	______
Predicted FIP I/O Scan Impact				______

Sweep Impact of Intelligent Option Modules

Intelligent option modules include Programmable Coprocessor Modules (PCM), Alphanumeric Display Coprocessor (ADC) Modules, Graphics Display Coprocessor (GDC) Modules, MAP LAN Modules, Ethernet LAN Modules, and Genius Bus Controllers being used for Genius LAN capabilities. The sweep impact for these intelligent option modules is highly variable. The opening of the system communications window and the polling of each module have relatively small impacts compared to the sweep impact of CPU memory read or write requests.

The following equations show how to calculate the fixed sweep of each module.

```
PCM          =  Polling Sweep Impact + Clock Refresh Impact once every 1/2 sec.

ADC          =  Polling Sweep Impact + Clock Refresh Impact once every 1/2 sec.

GDC          =  Polling Sweep Impact + Clock Refresh Impact once every 1/2 sec.

MAP LAN      =  Polling Sweep Impact + LAN I/O Scan Impact.

EthernetLAN  =  Polling Sweep Impact + LAN I/O Scan Impact.

GBC          =  Polling Sweep Impact + GBC I/O Scan Impact (see page A-13).

FBC          =  Polling Sweep Impact + FBC I/O Scan Impact (see prev section).
```

The table below shows the fixed sweep impact times in milliseconds for intelligent option modules. It also contains sweep impact times for reading and writing the PLCs system memories (includes all memories except %P and %L, which are slightly slower). The read and write service requests have two boundary conditions that change the times and are, therefore, broken up into three sets of times to reflect these boundary conditions.

Table A-12. Fixed Sweep Impact Times for Intelligent Option Modules *

Sweep Impact Item	924/ 925	914/ 915	781/782 788/789	731/732 771/772
Intelligent Option Modules				
First module (open comm window)				0.60
Per module (polling)				0.10
LAN module I/O Scan				0.15
PCM, ADC, GDC clock refresh				0.60
PLC Memory Access from IOMs				
Read/write 1 to 3 words. *				3.00
Read/write 4 to 128 words.				3.10
Read/write each additional 128 words.				1.00

* Times are in milliseconds. Not all of the timing information needed for the above table was available at print time for this manual (the blank spaces). Refer to the IPI for each CPU for this information.

** Reads can only fit 3 words into the basic message.
Writes can fit 4 words before a 256 byte text buffer is needed.

Note

Functions in bold type in Table A-10 on page A-17 impact the sweep continuously. All other functions impact the sweep only when invoked.

I/O Interrupt Performance and Sweep Impact

There are several important performance numbers for I/O interrupt blocks or Event-Triggered programs. The sweep impact of an I/O interrupt invoking an empty program or block measures the overall time of fielding the interrupt, starting up the program or block, exiting the program or block, and restarting the interrupted task. The maximum I/O interrupt rate reflects the limit of I/O interrupts invoking a minimal program or block at a sustained rate over time. The time to execute the logic contained in the interrupt program or block will affect the limit by causing the PLC to spend more time servicing I/O interrupts and thus reduce the maximum I/O interrupt rate.

The minimum, typical, and maximum interrupt response times reflect the time from when a single I/O module sees the input pulse until the first line of ladder logic or C code is executed in the I/O interrupt program or block. Minimum response time reflects a 300 μsec input card filter time + time from interrupt occurrence to first line of ladder logic in I/O interrupt program or block. The minimum response time can only be achieved when no intelligent option modules are present in the system and the programmer is not attached. Typical response time is the minimum response time plus a maximum interrupt latency of 2.0 ms for the model 731 CPU. This interrupt latency time is valid, except when one of the following operations occurs:

- The programmer is attached.
- A store of logic, ALT-S store, RUN mode store, or word-for-word change occurs.
- A fault condition (logging of a fault) occurs.
- Another I/O interrupt occurs.
- The CPU is transferring a large amount of input (or output) data from an I/O controller (e.g., a Genius Bus Controller or a FIP Bus Controller). Heavily loaded I/O controllers should be placed in the main rack whenever possible.

Any one of these events extends the interrupt latency (the time from when the interrupt card signals the interrupt to the CPU to when the CPU services the interrupt) beyond the typical value. However, the latency of an interrupt occurring during the processing of a preceding I/O interrupt is unbounded. I/O interrupts are processed sequentially so that the interrupt latency of a single I/O interrupt is affected by the duration of the execution time of all preceding interrupt blocks. (Worst case is that every I/O interrupt in the system occurs at the same time so that one of them has to wait for all others to complete before it starts.)

The maximum response time shown below does not include the two unbound events.

Table A-13. I/O Interrupt Block Performance and Sweep Impact Times *

Sweep Impact Item	924/ 925	914/ 915	781/782 788/789	731/732 771/772
I/O interrupt sweep impact				0.95
I/O interrupt response time				
Input card filter time				0.30
+ typical interrupt latency				1.40
+ interrupt to logic time				0.48
Minimum response time				0.78
Typical response time				2.18
Maximum response time				3.60
I/O interrupt rate limit				450 ints/sec
I/O interrupt rate limit w/no IOMs				750 ints/sec

* Times are in milliseconds. Not all of the timing information needed for the above table was available at print time for this manual (the blank spaces). Refer to the IPI for each CPU for this information.

Table A-14. Event-Triggered Interrupt Performance and Sweep Impact Times *

Sweep Impact Item	924/ 925	914/ 915	781/782 788/789
I/O interrupt sweep impact			
I/O interrupt response time			
Input card filter time			
+ typical interrupt latency			
+ interrupt to logic time			
Minimum response time			
Typical response time			
Maximum response time			
I/O interrupt rate limit			
I/O interrupt rate limit w/no IOMs			

* Times are in milliseconds. Not all of the timing information needed for the above table was available at print time for this manual (the blank spaces). Refer to the IPI for each CPU for this information.

The following form is a worksheet for the sweep impact times of programmer sweep impact, intelligent option modules, and I/O Interrupts. (Refer to tables A-4, A-10, and A-11.)

Note

Not all of the timing information needed for the above table was available at printing time for this manual (the blank spaces). Refer to the IPI for each CPU for this information.

Table A-15. Worksheet C: Programmer, IOM, I/O Interrupt Sweep Time

Programmer sweep impact		=	______
IOM - first module (open comm window)	______		
IOM - per module (polling)	+ ______		
LAN module I/O scan	+ ______		
Total IOM Sweep Impact		=	______
PLC memory access from IOMs		=	______
I.O interrupt sweep impact	______		
I/O interrupt response time	+ ______	=	______
Predicted Sweep Time (Other)		=	______

Timed Interrupt Performance

The sweep impact of a timed interrupt invoking an empty program block or Timed program measures the overall time of fielding the interrupt, starting up the program or block, exiting the program or block, and restarting the interrupted task. The minimum, typical, and maximum interrupt response times reflect the time from when a single timed interrupt occurs until the first line of ladder logic or C code is executed in the timed interrupt program or block. The minimum response time can only be achieved when no intelligent option modules are present in the system and the programmer is not attached. Typical response time is the minimum response time plus the CPU's maximum latency time. This interrupt latency time is valid, except when one of the following operations occurs:

- The programmer is attached.
- A store of logic, ALT-S store, RUN mode store, or word-for-word change occurs.
- A fault condition (logging of a fault) occurs.
- Another timed interrupt or I/O interrupt occurs.

Any one of these events extends the interrupt period beyond the typical value. However, the latency of an interrupt occurring during the processing of a preceding timed or I/O interrupt is unbounded. For interrupts, the worst case is that every timed and I/O interrupt in the system occurs at the same time so that one of them has to wait for all others to complete before it starts.

The maximum response time shown below does not include the two unbound events.

Table A-16. Timed Interrupt Performance and Sweep Impact Times *

Sweep Impact Item	924/ 925	914/ 915	781/782 788/789	731/732 771/772
Timed interrupt sweep impact				1.15
Timed interrupt response time				
Typical interrupt latency				1.40
+ interrupt to logic time				0.68
Minimum response time				0.68
Typical response time				2.08
Maximum response time				3.50

* Times are in milliseconds. Not all of the timing information needed for the above table was available at print time for this manual (the blank spaces). Refer to the IPI for each CPU for this information.

Table A-17. Event-Triggered Interrupt Performance and Sweep Impact Times *

Sweep Impact Item	924/ 925	914/ 915	781/782 788/789
Timed interrupt sweep impact			
Timed interrupt response time			
Typical interrupt latency			
+ interrupt to logic time			
Minimum response time			
Typical response time			
Maximum response time			

* Times are in milliseconds. Not all of the timing information needed for the above table was available at print time for this manual (the blank spaces). Refer to the IPI for each CPU for this information.

Examples of Calculating Predicted Sweep Times

The following two examples are provided to show how to calculate predicted sweep times. The first example is a small system and the second is a large system. Neither of these sweep time estimates include a time for logic execution. In both of these systems, the calculated sweep is for normal sweep time with point faults disabled, the PCM is idle, and the programmer is not attached. The times used in the calculation are extracted from tables A-3, A-5, A-7, A-8, and A-10. Sample forms for calculating predicted sweep times are provided after the examples.

Small System

PS	CPU 731	BTM	32PT Input	32PT Input	32PT Output	32PT Output	8CHN Analog Input	4CHN Analog Output	PCM
0	1	2	3	4	5	6	7	8	9

MAIN RACK

Sweep Calculations

$$\boxed{\text{Predicted Sweep}} = \boxed{\text{Base Sweep}} + \boxed{\text{I/O Scan Impact}} + \boxed{\text{PCM Impact}}$$

Base Sweep Time				2.00
I/O Scan Impact = I/O Scan Overhead + Series 90-70 I/O Scan Impact				
Number of discrete I/O modules — main rack		4		
Sweep impact time per discrete I/O module	x	0.08	=	0.32
Number of analog base and output modules — main rack		2		
Sweep impact time per analog base and output modules	x	0.08	=	0.16
I/O Scan Impact = 0.32 +16		0.48	=	0.48
I/O scan overhead		0.38	=	0.38
PCM Impact				
First module (open comm window)		0.60		
Per module (polling)	+	0.10	=	0.70
Predicted Sweep Time			=	4.04

Note

Times are in milliseconds.

Large System

PS	CPU 781	BTM	GBC 20Blks	GBC 20Blks	GBC 16Blks	GBC 16Blks	PCM	PCM	ENET LAN
0	1	2	3	4	5	6	7	8	9

MAIN RACK

PS	BRM	32PT Input	32PT Input	32PT Output	32PT Output	8CHN Analog Input	16CHN Analog Expndr	4CHN Analog Output	4CHN Analog Output
0	1	2	3	4	5	6	7	8	9

RACK1

PS	BRM	32PT Input	32PT Input	32PT Output	32PT Output	8CHN Analog Input	16CHN Analog Expndr	4CHN Analog Output	4CHN Analog Output
0	1	2	3	4	5	6	7	8	9

RACK2

The Genius block configuration used for this example is all 16 point grouped (QI) blocks with all bus addresses having contiguous reference addresses.

Predicted Sweep Calculations

$\boxed{\text{Predicted Sweep}} = \boxed{\text{Base Sweep}} + \boxed{\text{I/O Scan Impact}} + \boxed{\text{IOM Impact}}$

Base Sweep Time		0.10	=	0.10
I/O Scan overhead		0.20		
90-70 I/O scan impact (see table A-13, Worksheet A)	+	0.80		
Genius I/O scan impact (see table A-14, Worksheet B)	+	5.88		
I/O scan overhead			=	6.88
PCM impact	x	.30		
Number of PCMs		2		
		0.60		
LAN impact	+	.08		
IOM impact			=	.68
Predicted Sweep Time			=	8.66

Note

Times are in milliseconds.

Table A-18. Worksheet A

Number of expansion racks Sweep impact per expansion rack	x	2 .03	=	.06
Number of discrete I/O modules — main rack Sweep impact per discrete I/O module — main rack	x		=	
Number of discrete I/O modules — expansion rack Sweep impact per discrete I/O module — expansion rack	x	8 .05	=	.40
Number of analog input base and output modules — main rack Sweep impact per analog input base and output module — main rack	x	6 .04	=	.24
Number of analog input expander modules (same segment) — main rack Sweep impact per analog input expander modules (same segment) — main rack	x		=	
Number of analog input expander modules (new segment) — main rack Sweep impact per analog input expander modules (new segment) — main rack	x		=	
Number of analog input base and output modules — expansion rack Sweep impact per analog input base and output module — expansion rack	x		=	
Number of analog input base and output modules (same segment) — expansion rack Sweep impact per analog input base and output module (same segment) — expansion rack	x	2 .05	=	.10
Number of analog input base and output modules (new segment) — expansion rack Sweep impact per analog input base and output module (new segment) — expansion rack	x		=	
Predicted Series 90-70 I/O Module Sweep Impact				.80

Table A-19. Worksheet B

Open system communications window		0.30	=	0.30
GBC I/O scan		0.94		
GBC poll for background messages	x	0.04	=	0.98
Number of GBCs	x	4	=	3.92
Input block scan segments — number of		4		
Input block scan segments — sweep impact	x	0.02	=	0.08
Output block scan segments — number of		4		
Output block scan segments — sweep impact	x	0.02	=	0.08
Bytes of discrete I/O data on GBCs — main rack		288		
Sweep impact/byte,discrete I/O data — main rack	x	.0018	=	.5184
Bytes of discrete I/O data on GBCs — expansion racks				
Sweep impact/bytes of discrete I/O data — expansion racks	x		=	
Words of analog I/O data on GBCs — main rack				
Sweep impact/word analog I/O data — main rack	x		=	
Words of analog I/O data on GBCs — expansion racks				
Sweep impact/word analog I/O data — expansion racks	x		=	
Predicted Genius I/O Scan Impact				5.88

Table A-20. Sample Worksheet A

Number of expansion racks		______		
Sweep impact per expansion rack	x	______	=	______
Number of discrete I/O modules — main rack		______		
Sweep impact per discrete I/O module — main rack	x	______	=	______
Number of discrete I/O modules — expansion rack		______		
Sweep impact per discrete I/O module — expansion rack	x	______	=	______
Number of analog input base and output modules — main rack		______		
Sweep impact per analog input base and output module — main rack	x	______	=	______
Number of analog input expander modules (same segment) — main rack		______		
Sweep impact per analog input expander modules (same segment) — main rack	x	______	=	______
Number of analog input expander modules (new segment) — main rack		______		
Sweep impact per analog input expander modules (new segment) — main rack	x	______	=	______
Number of analog input base and output modules — expansion rack		______		
Sweep impact per analog input base and output module — expansion rack	x	______	=	______
Number of analog input base and output modules (same segment) — expansion rack		______		
Sweep impact per analog input base and output module (same segment) — expansion rack	x	______	=	______
Number of analog input base and output modules (new segment) — expansion rack		______		
Sweep impact per analog input base and output module (new segment) — expansion rack	x	______	=	______
Predicted Series 90-70 I/O Module Sweep Impact				______

Table A-21. Sample Worksheet B

Open system communications window		______	=	______
GBC I/O scan		______		
GBC poll for background messages	x	______	=	______
Number of GBCs	x	______	=	______
Input block scan segments — number of		______		
Input block scan segments — sweep impact	x	______	=	______
Output block scan segments — number of		______		
Output block scan segments — sweep impact	x	______	=	______
Bytes of discrete I/O data on GBCs — main rack		______		
Sweep impact/byte,discrete I/O data — main rack	x	______	=	______
Bytes of discrete I/O data on GBCs — expansion racks		______		
Sweep impact/bytes of discrete I/O data — expansion racks	x	______	=	______
Words of analog I/O data on GBCs — main rack		______		
Sweep impact/word analog I/O data — main rack	x	______	=	______
Words of analog I/O data on GBCs — expansion racks		______		
Sweep impact/word analog I/O data — expansion racks	x	______	=	______
Predicted Genius I/O Scan Impact				______

Table A-22. Sample Worksheet C

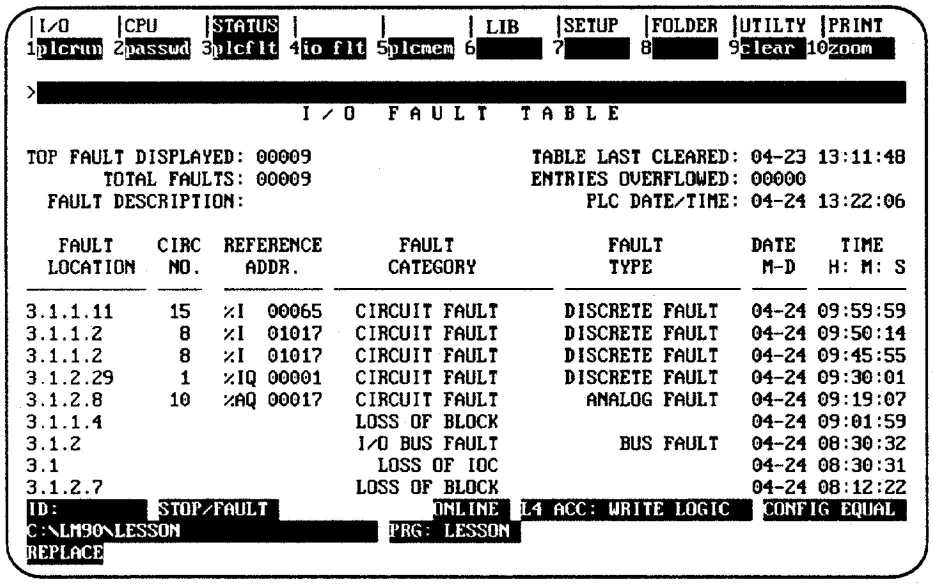
Programmer sweep impact		=	______
IOM — first module (open comm window)	______		
IOM — per module (polling)	+ ______		
LAN module I/O scan	+ ______		
Total IOM Sweep Impact		=	______
PLC memory access from IOMs		=	______
I.O interrupt sweep impact	______		
I/O interrupt response time	+ ______	=	______
Predicted Sweep Time (Other)		=	______

Appendix B

Interpreting Faults Using Logicmaster 90-70 Software

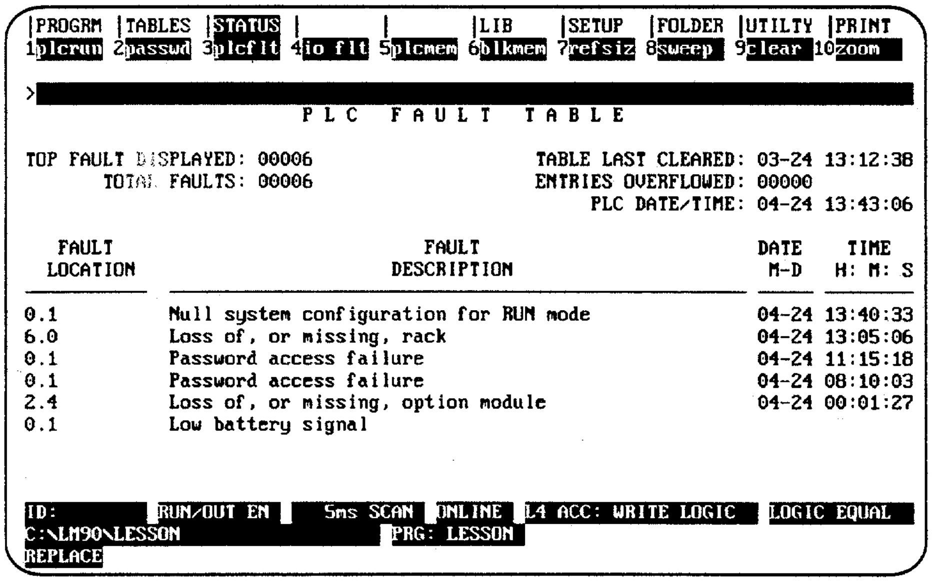
The Series 90-70 PLC maintains two fault tables, the I/O fault table for faults generated by I/O devices (including I/O controllers) and the PLC fault table for internal PLC faults. The information in this appendix will enable you to interpret the message structure format when reading these fault tables.

This is a sample I/O fault table, as displayed in Logicmaster 90-70 configuration software.



```
|I/O      |CPU      |STATUS  |        |        | LIB   |SETUP   |FOLDER  |UTILTY  |PRINT
1plcrun  2passwd  3plcflt  4io flt  5plcmem  6       7       8       9clear  10zoom

>
                             I / O   F A U L T   T A B L E

TOP FAULT DISPLAYED: 00009                               TABLE LAST CLEARED: 04-23 13:11:48
       TOTAL FAULTS: 00009                               ENTRIES OVERFLOWED: 00000
  FAULT DESCRIPTION:                                          PLC DATE/TIME: 04-24 13:22:06

   FAULT      CIRC  REFERENCE       FAULT               FAULT          DATE    TIME
  LOCATION     NO.    ADDR.        CATEGORY              TYPE           M-D    H: M: S

3.1.1.11       15   %I  00065   CIRCUIT FAULT       DISCRETE FAULT    04-24 09:59:59
3.1.1.2         8   %I  01017   CIRCUIT FAULT       DISCRETE FAULT    04-24 09:50:14
3.1.1.2         8   %I  01017   CIRCUIT FAULT       DISCRETE FAULT    04-24 09:45:55
3.1.2.29        1   %IQ 00001   CIRCUIT FAULT       DISCRETE FAULT    04-24 09:30:01
3.1.2.8        10   %AQ 00017   CIRCUIT FAULT         ANALOG FAULT    04-24 09:19:07
3.1.1.4                         LOSS OF BLOCK                         04-24 09:01:59
3.1.2                           I/O BUS FAULT              BUS FAULT  04-24 08:30:32
3.1                               LOSS OF IOC                         04-24 08:30:31
3.1.2.7                         LOSS OF BLOCK                         04-24 08:12:22
ID:           STOP/FAULT                 ONLINE  L4 ACC: WRITE LOGIC   CONFIG EQUAL
C:\LM90\LESSON                          PRG: LESSON
REPLACE
```

This is a sample PLC fault table, as displayed in Logicmaster 90-70 programming software.



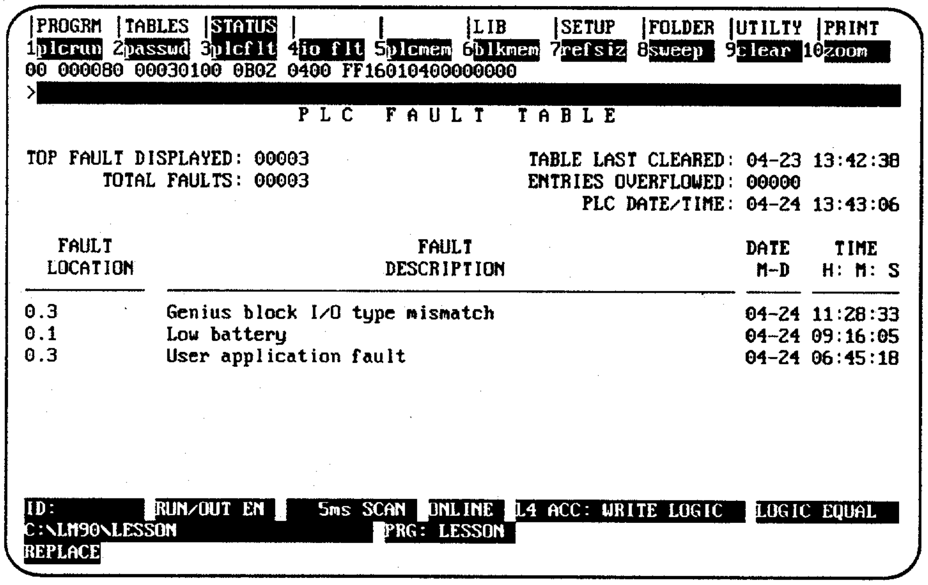
```
|PROGRM  |TABLES  |STATUS  |        |        |LIB     |SETUP   |FOLDER  |UTILTY  |PRINT
1plcrun  2passwd  3plcflt  4io flt  5plcmem  6blkmem  7refsiz  8sweep  9clear  10zoom

>
                             P L C   F A U L T   T A B L E

TOP FAULT DISPLAYED: 00006                               TABLE LAST CLEARED: 03-24 13:12:38
      TOTAL FAULTS: 00006                                ENTRIES OVERFLOWED: 00000
                                                              PLC DATE/TIME: 04-24 13:43:06

   FAULT                          FAULT                                    DATE    TIME
  LOCATION                     DESCRIPTION                                  M-D    H: M: S

0.1            Null system configuration for RUN mode                     04-24 13:40:33
6.0            Loss of, or missing, rack                                  04-24 13:05:06
0.1            Password access failure                                    04-24 11:15:18
0.1            Password access failure                                    04-24 08:10:03
2.4            Loss of, or missing, option module                         04-24 00:01:27
0.1            Low battery signal

ID:           RUN/OUT EN     5ms SCAN   ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                         PRG: LESSON
REPLACE
```

Both tables contain similar information.

- The PLC fault table contains:
  - Fault location.
  - Fault description.
  - Date and time of fault.
- The I/O fault table contains:
  - Fault location.
  - Circuit number.
  - Reference address.
  - Fault category.
  - Fault type.
  - Date and time of fault.

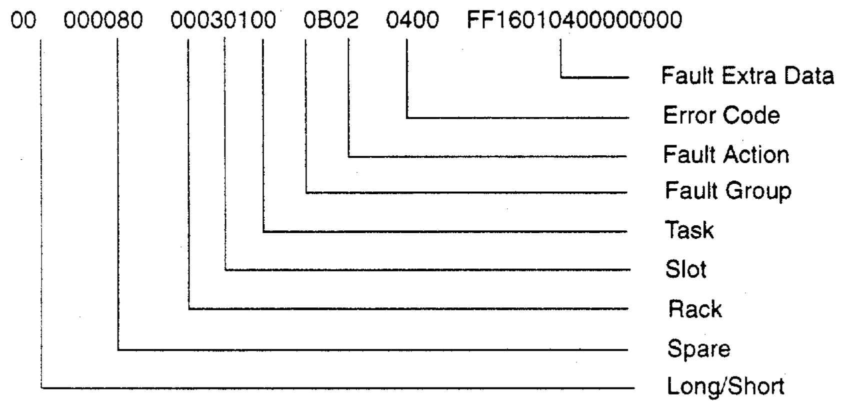
The Series 90-70 PLC maintains additional information on each fault that, because of space limitations on the Logicmaster screen, is not displayed. This additional fault table information may be viewed by positioning the cursor on the fault and pressing the **CTRL** and **F** keys together. A line of hexadecimal characters is displayed on the line directly beneath the softkey menus. This is the full fault entry, as stored by the PLC CPU. This additional data, along with suggestions for fixing the fault, may also be viewed by positioning the cursor on the fault and pressing the **Zoom (F10)** key.

The following screen is an example display of the PLC fault table. Note the hexadecimal information displayed on the line beneath the softkey menus.


```
|PROGRM |TABLES |STATUS |        |        |LIB     |SETUP   |FOLDER  |UTILTY  |PRINT
1plcrun 2passwd 3plcflt 4io flt 5plcmem 6blkmem 7refsiz 8sweep  9clear  10zoom
00 000080 00030100 0B02 0400 FF16010400000000
>
                          P L C   F A U L T   T A B L E

TOP FAULT DISPLAYED: 00003                         TABLE LAST CLEARED: 04-23 13:42:38
      TOTAL FAULTS: 00003                         ENTRIES OVERFLOWED: 00000
                                                       PLC DATE/TIME: 04-24 13:43:06

  FAULT                             FAULT                              DATE    TIME
 LOCATION                        DESCRIPTION                           M-D    H: M: S

0.3           Genius block I/O type mismatch                          04-24 11:28:33
0.1           Low battery                                             04-24 09:16:05
0.3           User application fault                                  04-24 06:45:18





ID:            RUN/OUT EN     5ms SCAN  ONLINE  L4 ACC: WRITE LOGIC    LOGIC EQUAL
C:\LM90\LESSON                          PRG: LESSON
REPLACE
```

PLC Fault Table

The following diagram identifies each field in the fault entry:



The following paragraphs describe each field in the fault entry. Included are tables describing the range of values each field may have.

Long/Short Indicator

This byte indicates whether the fault contains 8 bytes or 24 bytes of fault extra data.

Type	Code	Fault Extra Data
Short	00	8 bytes
Long	01	24 bytes

Spare

These six bytes are pad bytes, used to make the PLC fault table entry exactly the same length as the I/O fault table entry.

Rack

The rack number ranges from 0 to 7. Zero is the main rack, containing the PLC. Racks 1 through 7 are expansion racks, connected to the PLC through a Bus Transmitter Module in the main rack and Bus Receiver Modules in the expansion racks.

Slot

The slot number ranges from 0 to 9. The PLC CPU always occupies slot 1 in the main rack (rack 0).

Task

The task number ranges from 0 to +65,535. Sometimes the task number provides additional information to PLC engineers; typically, however, the task number can be ignored.

PLC Fault Group

Fault group is the highest classification of a fault. It identifies the general category of the fault. The fault description text displayed by Logicmaster 90-70 software is based on the fault group and the error codes.

Table B-1 lists the possible fault groups in the PLC fault table. Group numbers less than 80 (Hex) are maskable faults, while group numbers greater than or equal to 80 (Hex) are non-maskable faults.

The last non-maskable fault group, ***Additional PLC Fault Codes,*** is declared for the handling of new fault conditions in the system without the PLC having to specifically know the alarm codes. All unrecognized PLC-type alarm codes belong to this group.

Table B-1. PLC Fault Groups

Group Number			
Decimal	Hexadecimal	Group Name	Fault Action
1	1	Loss of, or missing, rack.	Fatal
4	4	Loss of, or missing, option module.	Diagnostic
5	5	Addition of, or extra, rack.	Diagnostic
8	8	Addition of, or extra, option module.	Diagnostic
11	B	System configuration mismatch.	Fatal
12	C	System bus error.	Diagnostic
13	D	PLC CPU hardware failure. *	Fatal
14	E	Non-fatal module hardware failure.	Diagnostic
16	10	Option module software failure.	Diagnostic
17	11	Program block checksum failure.	Fatal
18	12	Low battery signal.	Diagnostic
19	13	Constant sweep time exceeded.	Diagnostic
20	14	PLC system fault table full.	Diagnostic
21	15	I/O fault table full.	Diagnostic
22	16	User Application fault.	Diagnostic
–	–	Additional PLC fault codes.	As specified
128	80	System bus failure.	Fatal
129	81	No user's program on power-up.	Informational
130	82	Corrupted user RAM detected.	Fatal
131	83	Window completion failure in Constant Sweep mode (i.e., all windows failed to receive their allotted time).	Informational
132	84	Password access failure.	Informational
134	86	Null system configuration for RUN mode.	Informational
135	87	PLC CPU software failure.	Fatal
136	88	More than the allowable number of I/O bus controllers were found in the system.	Fatal
137	89	PLC sequence-store failure.	Fatal

* The PLC OK LED will flash on and off to indicate that the failure was not serious enough to prevent programmer communications from retrieving the fault table information.

PLC Fault Action

Each fault may have one of three actions associated with it.

Table B-2. PLC Fault Actions

Fault Action	Action Taken by CPU	Code
Informational	Log fault in fault table.	1
Diagnostic	Log fault in fault table. Set fault references.	2
Fatal	Log fault in fault table. Set fault references. Go to StOP mode.	3

Error Code

The error code further describes the fault. Each fault group has its own set of error codes.

The first table below shows error codes for the PLC Software Error Group (Group 87H).

Table B-3. Alarm Error Codes for PLC CPU Software Faults

Decimal	Hexadecimal	Name
20	14	Corrupted PLC Program Memory.
39	27	Corrupted PLC Program Memory.
82	52	Backplane Communications Failed.
90	5A	User Shut Down Requested.
123	7B	Remote I/O Scanner Communications Failure.
149	95	Store from Flash on Power-Up Failed. __Note__: The first byte of the Fault Specific Data describes why the store from flash failed: __Error__ / __Fault Specific Data Value__ / __Description__ DEVICE_NOT_AVAILABLE / CF / Specific device is not available in the system. BAD_DEVICE_DATA / CC / Data stored on device has been corrupted and is no longer reliable. Or, Flash Memory has not been initialized. DEVICE_RW_ERROR / CB / Error occurred during a read/write of the Flash Memory device. FLASH_INCOMPAT_ERROR / 8E / Data in Flash Memory is incompatible with the PLC CPU firmware release due to the CPU firmware revision numbers, the instruction groups supported, or the CPU model number. ITEM_NOT_FOUND_ERROR / 8D / One or more specified items were not found in Flash Memory.
All others		PLC CPU Internal System Error.

The next table shows the error codes for all the other fault groups.

Table B-4. Alarm Error Codes for PLC CPU Faults

Decimal	Hexadecimal	Name
PLC Error Codes for Loss of Option Module Group		
3	3	Bus Transmitter Module Found in Expansion Rack.
22	16	Analog Expander Located to Left of Base Converter Module.
25	19	Lost Analog Expander Module.
44	2C	Option Module Soft Reset Failed.
45	2D	Option Module Soft Reset Failed.
59	3B	Loss of, or missing communications driver.
60	3C	Module in firmware update mode.
65	41	Module is in Standalone mode; mail system not initialized.
255	FF	Option Module Communication Failed.
Error Codes for Addition of, or Extra Rack Group		
1	1	Addition of, or Extra Rack .
Error Codes for Reset of, Addition of, or Extra Option Module Group		
2	2	Module Restart Complete.
3	3	LAN Interface Restart Complete; Running a Utility.
All others		Reset of, Addition of, or Extra Option Module.
Error Codes for Module Hardware Error Group		
1	1	Non-fatal LAN Hardware error.
416	1A0	Required 12V PS failed or missing.
450	1C2	LAN Controller Underrun/Overrun; Resuming.
451	1C3	LAN Interface Failure; Switched Off Network.
452	1C4	LAN Network Problem; Performance Degraded.
453	1C5	LAN Severe Network Problem; Attempting Recovery.
454	1C6	LAN Transceiver Fault; Off Network Until Fixed.
Error Codes for Option Module Software Failure Group		
1	1	Unsupported Board Type.
2	2	COMREQ — mailbox full on outgoing message that starts the COMREQ.
3	3	COMREQ — mailbox full on response.
4	4	More Than One BTM in Rack.
5	5	Backplane Communications with PLC; Lost Request.
10	A	Error with LAN interaction.
11	B	Resource (alloc, tbl ovrflw, etc.) error.
12	C	VME backplane error.
13	D	User program error.
401	191	Module Software Corrupted; Requesting Reload.
402	192	LAN System Software Fault; Resuming.
403	193	LAN System Software Fault; Aborted Assoc and Resuming.
404	194	LAN System Software Fault; Restarted LAN I/F.
405	195	LAN System Software Fault; Reinitializing LLC.

Table B-4. Alarm Error Codes for PLC CPU Faults (cont'd)

Decimal	Hexadecimal	Name
		Error Codes for System Configuration Mismatch Group
2	2	Genius Block Model Number Mismatch.
4	4	Genius Block I/O Type Mismatch.
7	7	Daughter Board Mismatch.
8	8	Analog Expansion Mismatch.
9	9	Genius Block Broadcast Control Data (BCD) Length Mismatch.
10	A	Unsupported Feature.
11	B	Revision A BTM not in Right Slot.
14	E	LAN Duplicate MAC Address.
15	F	LAN Duplicate MAC Address Resolved.
16	10	LAN MAC Address Mismatch.
17	11	LAN Soft Switch/Modem Mismatch.
19	13	Genius Block Direct Control Data (DCD) Length Mismatch.
23	17	Program exceeds memory limits.
29	1D	Incompatible scheduling mode.
30	1E	Reference length mismatch.
31	1F	Invalid configuration parameters.
32	20	New configuration requires reset.
36	24	I/O specification mismatch.
37	25	Controller reference out of range.
39	27	Bad interrupt trigger.
		Error Codes for System Bus Error Group
4	4	Unrecognized VME Interrupt Error.
All others		System Bus Error.
		Error Codes for Program Block Checksum Group
0	0	Corruption of program block header in the Series 90-70 PLC.
1	1	Corruption of stored OMF records stored in Series 90-70 PLC.
2	2	Corruption of stored OMF records stored in Series 90-70 PLC.
3	3	Program or program block checksum failure.
		Error Codes for Low Battery Signal
0	0	Failed battery on PLC CPU or other module.
1	1	Low battery on PLC CPU or other module.
2	2	Failed battery from VME backplane.
3	3	Low battery from VME backplane.

Table B-4. Alarm Error Codes for PLC CPU Faults (cont'd)

Decimal	Hexadecimal	Name
		Error Codes for User Application Fault Group
1	1	Indirect Reference Address Out of Range.
2	2	PLC Watchdog Timer Timed Out.
3	3	GBC COMREQ.
4	4	GBC Bkgnd msg – Bad Genius Bus Request.
5	5	COMREQ – WAIT mode not available for this command.
6	6	COMREQ – Bad Task ID.
7	7	Application Stack Overflow.
8	8	LAN Data Memory Exhausted – Check Parms; Resuming.
9	9	Bad Remote Application Request; Discarded Request.
10	A	Bad Local Application Request; Discarded Request.
11	B	LAN I/F Capacity Exceeded; Discarded Request.
12	C	LAN PROM/Software Mismatch; Running Soft Sw Util.
13	D	LAN I/F Can't Init – Check Parms; Running Soft Sw Util.
14	E	Run-time error detected in an external block.
15	F	SORT function in an interrupt did not execute.
17	11	Standalone Run-Time Error.
28	1C	Program Exceeded Wind Down.
29	1D	Program Not Readied.
		Error Codes for System Bus Failure Group
1	1	Operating system.
		Error Codes for Corrupted User RAM on Powerup Group
1	1	Corrupted User RAM on Power-up.
2	2	Illegal Boolean Opcode Detected.
5	5	Partial Store failure on second pass of parsing OMF.
6	6	Corrupted Remote I/O Scanner EEPROM; Config Lost.
		Error Codes for PLC CPU Hardware Faults
2169	879	Remote I/O Scanner Comm Failure; Verify Bus.
2172	87C	Remote I/O Scanner Serial Bus Address Conflict.
2048 to 4095	800 to FFF	Remote I/O Scanner Hardware Fault.
All other codes		PLC CPU Hardware Failure.

PLC Fault Extra Data

The *Fault Extra Data* field in the PLC fault table contains details of the fault entry. Some examples of what data may be present are:

System Configuration Mismatch

Four error codes in the System Configuration Mismatch group supply fault extra data:

Table B-5. PLC Fault Extra Data – System Configuration Mismatch

Fault Extra Data Byte Number	Model Number Mismatch
[0]	FF.
[1]	Bus address.
[2]	Installed module's model number.
[3]	Configured model number.
Fault Extra Data Byte Number	**I/O Type Mismatch**
[0]	FF.
[1]	Bus address.
[2]	Installed module's I/O type.
[3]	Configured module's I/O type.
Fault Extra Data Byte Number	**BCD Length Mismatch**
[0]	FF.
[1]	Bus address.
[2]	Module's broadcast data length.
[3]	Configured module's broadcast data length.
Fault Extra Data Byte Number	**DCD Length Mismatch**
[0]	FF.
[1]	Bus address.
[2]	Module's directed data length.
[3]	Configured module's directed data length.

The following table shows the Genius model numbers, used when a model number mismatch occurs.

Table B-6. Genius Block Model Numbers

Number		
Decimal	Hexadecimal	Description
4	4	Genius Network Interface (GENI).
5	5	Phase B Hand Held Monitor.
6	6	Phase B Series Six Genius Bus Controller with Diagnostics.
7	7	Phase B Series Six Genius Bus Controller without Diagnostics.
8	8	PLCM/Series Six.
9	9	PLCM/Series 90-40.
10	A	Series 90-70 Single Channel Bus Controller.
11	B	Series 90-70 Dual Channel Bus Controller.
12	C	Series 90-10 Genius Communications Module.
13	D	Series 90-30 Genius Communications Module.
32	20	High Speed Counter.
69	45	Phase B 115Vac 8-point (2 amp) Grouped Block.
70	46	Phase B 115Vac/125Vdc 8-point Isolated Block.
70	46	Phase B 115Vac/125Vdc 8-point Isolated Block without Failed Switch.
71	47	Phase B 220Vac 8-point Grouped Block.
72	48	Phase B 24-48Vdc 16-point Proximity Sink Block.
72	48	Phase B 24Vdc 16-point Proximity Sink Block.
73	49	Phase B 24-48Vdc 16-point Source Block.
73	49	Phase B 24Vdc 16-point Proximity Source Block.
74	4A	Phase B 12-24Vdc 32-point Sink Block.
75	4B	Phase B 12-24Vdc 32-point Source Block.
76	4C	Phase B 12-24Vdc 32-point 5V Logic Block.
77	4D	Phase B 115Vac 16-point Quad State Input Block.
78	4E	Phase B 12-24Vdc 16-point Quad State Input Block.
79	4F	Phase B 115/230Vac 16-point Normally Open Relay Block.
80	50	Phase B 115/230Vac 16-point Normally Closed Relay Block.
81	51	Phase B 115Vac 16-point AC Input Block.
82	52	Phase B 115Vac 8-point Low-Leakage Grouped Block.
127	7F	Genius Network Adapter (GENA).
131	83	Phase B 115Vac 4-input, 2-output Analog Block.
132	84	Phase B 24Vdc 4-input, 2-output Analog Block.
133	85	Phase B 220Vac 4-input, 2-output Analog Block.
134	86	Phase B 115Vac Thermocouple Input Block.
135	87	Phase B 24Vdc Thermocouple Input Block.
136	88	Phase B 115Vac RTD Input Block.
137	89	Phase B 24/48Vdc RTD Input Block.
138	8A	Phase B 115Vac Strain Gauge/mV Analog Input Block.
139	8B	Phase B 24Vdc Strain Gauge/mV Analog Input Block.
140	8C	Phase B 115Vac 4-input, 2-output Current Source Analog Block.
141	8D	Phase B 24Vdc 4-input, 2-output Current Source Analog Block.

If the model number is 7FH (Genius Network Adapter), the block may be one of the following. (The GENA Application ID is shown for reference.)

Table B-7. GENA Application ID Numbers

Number		
Decimal	Hexadecimal	Description
131	83	115Vac/230Vac/125Vdc Power Monitor Module.
132	84	24/48Vdc Power Monitor Module.
160	A0	Genius Remote 90-70 Rack Controller.

When the system configuration mismatch is an I/O type mismatch, the installed module I/O type is one of the following:

Table B-8. Genius Installed Module I/O Types

Value	Description
01	Input only.
02	Output only.
03	Combination.

When the system configuration mismatch is an I/O type mismatch, the configured module I/O type is one of the following. (All values are in hexadecimal.)

Table B-9. Genius Configured Module I/O Types

Value		
Decimal	Hexadecimal	Description
0	0	Discrete input.
1	1	Discrete output.
2	2	Analog input.
3	3	Analog output.
4	4	Discrete grouped.
5	5	Analog grouped.
20	14	Analog in, discrete in.
21	15	Analog in, discrete out.
24	18	Analog in, discrete grouped.
30	1E	Analog out, discrete in.
31	1F	Analog out, discrete out.
34	22	Analog out, discrete grouped.
50	32	Analog grouped, discrete in.
51	33	Analog grouped, discrete out.
54	36	Analog grouped, discrete grouped.

Program Block Checksum Failure

The name of the offending program block is contained in the first eight bytes of the ***Fault Specific Data*** field.

PLC CPU Hardware Failure (RAM Failure)

For a RAM failure in the PLC CPU (one of the faults reported as a PLC CPU hardware failure), the address of the failure is stored in the first four bytes of the field.

Application Fault

Indirect Reference Overflow: The offset address of where the call was made is located in the first two bytes. The offending reference (segment selector and offset) is located in the next four bytes. The name of the program block in which the function call resides is contained in the next eight bytes.

Bad COMREQ Status Pointer: The first byte contains a hex FF. The next four bytes contain the segment selector and offset of the status pointer into which the PLC CPU could not write.

Bad Genius Bus Request: Four bytes are used, unless the request is a Read or Write Device. In these two datagrams, eight bytes are used.

Table B-10. Fault Specific Data - Bad Genius Bus Request

Fault Specific Data	Bad Genius Bus Request
[0]	FF.
[1]	Bus address of requestor.
[2]	Function code.
[3]	Subfunction code.
[4]	Segment selector, if Read/Write device.
[5]	LSB of offset, if Read/Write device.
[6]	MSB of offset, if Read/Write device.
[7]	Length, if Read/Write device.

PLC Fault Time Stamp

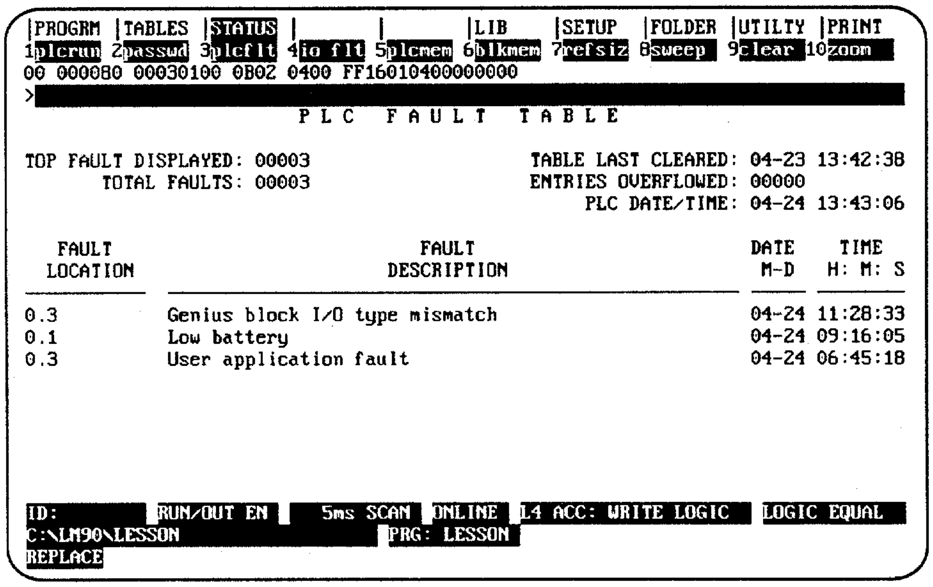
The six-byte time stamp is the value of the system clock when the fault was recorded by the PLC CPU. (Values are coded in BCD format.)

Table B-11. PLC Fault Time Stamp

Byte Number	Description
1	Seconds.
2	Minutes.
3	Hours.
4	Day of the month.
5	Month.
6	Year.

PLC Fault Table Examples

The following sample PLC fault table displays the first fault above the command line. In this section, each of these three faults is examined in detail.



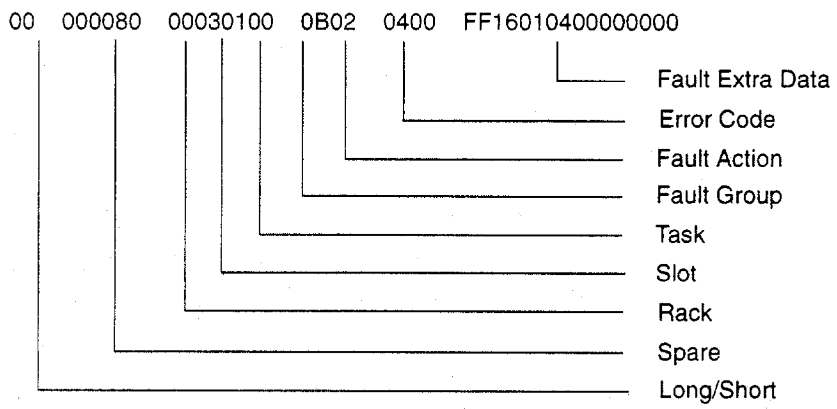
```
|PROGRM |TABLES |STATUS |        |        |LIB     |SETUP   |FOLDER  |UTILTY  |PRINT
1plcrun 2passwd 3plcflt 4io flt 5plcmem 6blkmem 7refsiz 8sweep  9clear  10zoom
00 000080 00030100 0B02 0400 FF16010400000000
>
                        P L C   F A U L T   T A B L E

TOP FAULT DISPLAYED: 00003                        TABLE LAST CLEARED: 04-23 13:42:38
       TOTAL FAULTS: 00003                        ENTRIES OVERFLOWED: 00000
                                                      PLC DATE/TIME: 04-24 13:43:06

   FAULT                          FAULT                              DATE    TIME
  LOCATION                     DESCRIPTION                            M-D   H: M: S

0.3          Genius block I/O type mismatch                          04-24 11:28:33
0.1          Low battery                                             04-24 09:16:05
0.3          User application fault                                  04-24 06:45:18

ID:          RUN/OUT EN    5ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                       PRG: LESSON
REPLACE
```

Genius Block I/O Type Mismatch Example

The Genius Block I/O Type Mismatch fault entry is explained below. (All data is in hexadecimal.)

Field	Value	Description
Long/Short	00	This fault contains 8 bytes of fault extra data.
Rack	00	Main rack (rack 0).
Slot	03	Slot 3. In this configuration, slot 3 contains a Genius Bus Controller.
Task	01	Single channel bus controller has only one task.
Fault Group	0B	System Configuration Mismatch fault.
Fault Action	02	Diagnostic fault.
Error Code	04	Error code 04 in the System Configuration Mismatch group is a Genius Block I/O Type Mismatch.
Fault Extra Data		IO Type Mismatch error has four bytes of fault extra data.
[0]	FF	Flag byte.
[1]	16	Serial bus address: 16 hex is 22 decimal.
[2]	01	Installed module type. 01 is an inputs only module.
[3]	04	Configured module I/O type. 04 is a combination module.

The configuration file stored from Logicmaster 90-70 software shows that the device at serial bus address 22 is a combination (mixed) module. However, the "Read ID Reply" message the Genius Bus Controller received from the device at serial bus address 22 shows that the device is configured for inputs only. The Genius Bus Controller logged this fault when it detected the mismatch.

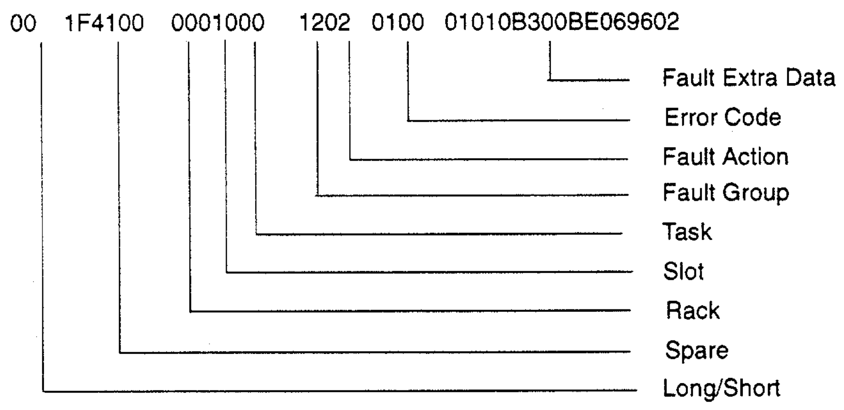


Low Battery Signal Example

In this sample screen, placing the cursor on the low battery signal and pressing **CTRL-F** displays the data shown below. (All data is in hexadecimal.)

Field	Value	Description
Long/Short	00	This fault contains 8 bytes of fault extra data.
Rack	00	Main rack (rack 0).
Slot	01	Slot 1. In all configurations, slot 1 in rack 0 contains the PLC CPU.
Task	00	PLC CPU.
Fault Group	12	Low battery signal.
Fault Action	02	Diagnostic fault.
Error Code	01	Error code 01 in the Low Battery group is one of four low battery conditions detected by the PLC CPU.
Fault Extra Data		No fault extra data for a low battery signal.

This fault occurred because the PLC CPU detected a low battery signal.

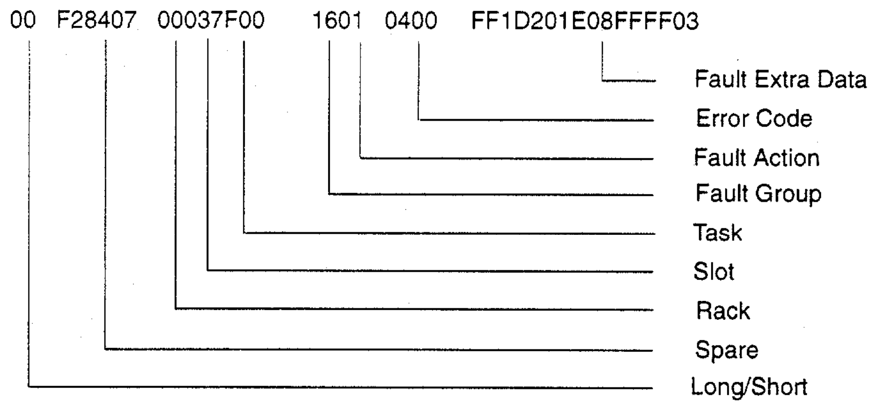


User Application Fault Example

Moving the cursor to the User Application fault in this example and pressing **CTRL-F** displays the data shown below. (All data is in hexadecimal.)

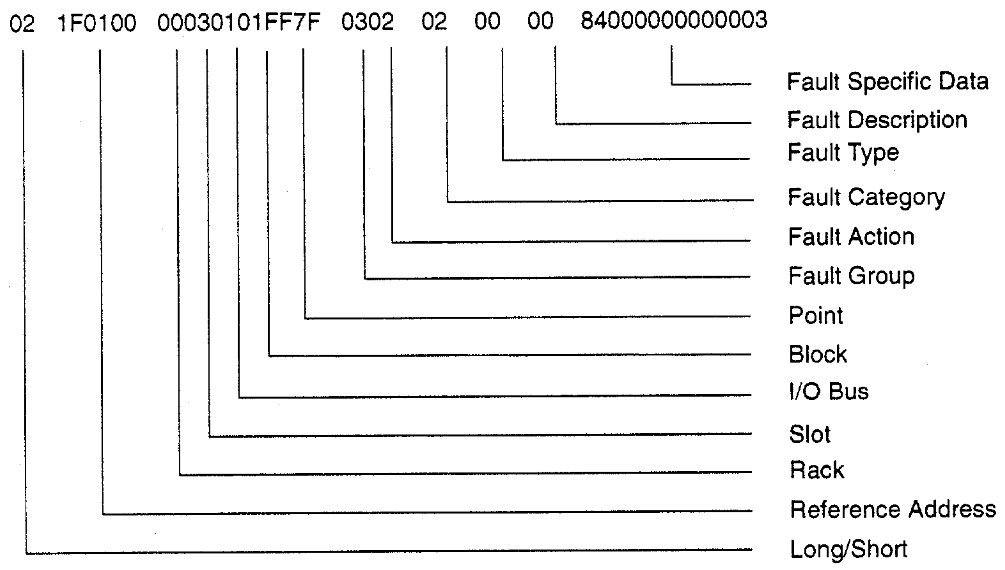
Field	Value	Description
Long/Short	00	This fault contains 8 bytes of fault extra data.
Rack	00	Main rack (rack 0).
Slot	03	Slot 3. In this configuration, slot 3 contains a Genius Bus Controller.
Task	7F	When the GBC registers a User Application fault in the PLC fault table, it places a 7F in the task byte of the fault.
Fault Group	16	User Application fault.
Fault Action	01	Informational fault.
Error Code	04	Error code 04 in the Application Fault group is a Bad Genius Bus Request. This fault occurs when the Genius Bus Controller receives a Read or Write Device datagram from another device on the Genius Bus that cannot be successfully completed.
Fault Extra Data		Bad Genius Bus Request error has eight bytes of fault extra data.
[0]	FF	Flag byte.
[1]	1D	Serial bus address: 1D hex is 29 decimal.
[2]	20	Function code in the datagram: a GE Fanuc datagram.
[3]	1E	Subfunction code in the datagram: a Read Device.
[4]	08	Segment selector: 08 is %R memory.
[5]	FF	Least significant byte of offset.
[6]	FF	Most significant byte of offset.
[7]	03	Length of data to read: 3 words.

The Genius Bus Controller received a Read Device datagram from serial bus address 29, which requested three words of %R memory be read starting at %R65536. Since this is beyond the range of the largest value %R can have, the bus controller registered an informational user application fault in the PLC fault table.



I/O Fault Table

The following diagram identifies the hexadecimal information displayed in each field in the fault entry.



The following paragraphs describe each field in the I/O fault table. Included are tables describing the range of values each field may have.

Long/Short Indicator

This byte indicates whether the fault contains 5 bytes or 21 bytes of fault specific data.

Table B-12. I/O Fault Table Format Indicator Byte

Type	Code	Fault Specific Data
Short	02	5 bytes
Long	03	21 bytes

Reference Address

Reference address is a three-byte address containing the I/O memory type and location (or offset) in that memory which corresponds to the point experiencing the fault. Or, when a Genius block fault or integral analog module fault occurs, the reference address refers to the first point on the block where the fault occurred.

Table B-13. I/O Reference Address

Byte	Description	Range
0	Memory Type	0 - FF
1-2	Offset	0 - 12K (decimal)

The memory type byte is one of the following values.

Table B-14. I/O Reference Address Memory Type

Name	Value (Hexadecimal)
Analog input	0A
Analog output	0C
Analog grouped	0D
Discrete input	10 or 46
Discrete output	12 or 48
Discrete grouped	1F

I/O Fault Address

The I/O fault address is a six-byte address containing rack, slot, bus, block, and point address of the I/O point which generated the fault. The point address is a word; all other addresses are one byte each. All five values may not be present in a fault.

When an I/O fault address does not contain all five addresses, a 7F hex appears in the address to indicate where the significance stops. For example, if 7F appears in the bus byte, then the fault is a module fault. Only rack and slot values are significant.

Rack

The rack number ranges from 0 to 7. Zero is the main rack, i.e., the one containing the PLC. Racks 1 through 7 are expansion racks, connected to the PLC through a Bus Transmitter Module in the main rack and Bus Receiver Modules in the expansion racks.

Slot

The slot number ranges from 0 to 9. The PLC CPU always occupies slot 1 in the main rack (rack 0).

I/O Bus

The I/O bus number ranges from 0 to 15. When the module in the slot is a single-channel Genius Bus Controller, this number is always one. When the module is an integral analog module, this designates which expansion channel generated the fault.

Block

Block refers to the serial bus address of the Genius block which reported or has the fault.

Point

Point ranges from 1 to 1024 (decimal). It tells which point on the block has the fault when the fault is a point-type fault.

I/O Fault Group

Fault group is the highest classification of a fault. It identifies the general category of the fault. The fault description text displayed by Logicmaster 90-70 software is based on the fault group and the error codes.

Table B-15 lists the possible fault groups in the I/O fault table. Group numbers less than 80 (Hex) are maskable faults.

The last non-maskable fault group, ***Additional I/O Fault Codes***, is declared for the handling of new fault conditions in the system without the PLC having to specifically know the alarm codes. All unrecognized I/O-type alarm codes belong to this group.

Table B-15. I/O Fault Groups

Group Number	Group Name	Fault Action
2	Loss of, or missing, IOC.	Fatal
3	Loss of, or missing, I/O module.	Diagnostic
6	Addition of, or extra, IOC.	Diagnostic
7	Addition of, or extra, I/O module.	Diagnostic
9	IOC or I/O bus fault.	Diagnostic
A	I/O module fault.	Diagnostic
F	IOC software failure.	Fatal
–	Additional I/O fault codes.	As specified

I/O Fault Action

The fault action specifies what action the PLC CPU should take when a fault occurs. The following table lists possible fault actions.

Table B-16. PLC Fault Actions

Fault Action	Action Taken by CPU	Code
Informational	Log fault in fault table.	1
Diagnostic	Log fault in fault table. Set fault references.	2
Fatal	Log fault in fault table. Set fault references. Go to STOP mode.	3

I/O Fault Category

The I/O fault category specifies a general classification of the fault. It is similar to the I/O fault group, discussed earlier. I/O fault categories are listed in the following table.

Table B-17. I/O Fault Categories

Decimal Number	Hex Code	Description
1	1	Circuit fault: Short circuit, open wire, etc.
2	2	Loss of block: Block no longer responding.
3	3	Addition of block: New block appeared.
4	4	Unused category.
5	5	Unused category.
6	6	Genius bus fault.
7	7	Global memory fault.
8	8	EEPROM fault, watchdog timeout.
9	9	Addition of IOC.
10	A	Loss of, or missing, IOC.
11	B	IOC software fault.
12	C	Forced circuit: A Genius I/O point was forced with the HHM.
13	D	Unforced circuit: HHM force was removed.
14	E	Loss of Series 90 integral card.
15	F	Addition of Series 90 integral card.
16	10	Found extra Series 90 integral card.
17	11	Found extra Genius block.
18	12	An IOC detected a hardware failure or a baud rate mismatch.
19	13	Genius bus controller has stopped reporting faults because too many faults have occurred.
20	14	Configuration mismatch fault for I/O modules.
21	15	GBC software exception.
22	16	Redundant Genius block switched buses.
23	17	Block not active on redundant bus.

I/O Fault Type

The I/O fault type component creates sub-categories under the circuit fault, module fault, I/O bus fault, loss of block, excessive faults, I/O configuration mismatch, and GBC software exception categories. It is undefined for other fault categories, but is always set to zero when the fault category is something other than these seven categories. Table B-18 lists the I/O fault types.

Table B-18. I/O Fault Types

Number	Description
	I/O Fault Types for the Circuit Fault Category
1	Circuit fault on a discrete I/O point.
2	Circuit fault on an analog I/O channel.
3	Fault on a GENA.
4	Fault on a low-level analog input channel.
5	Fault on Remote I/O Scanner
	I/O Fault Types for the Module Fault Category
0	Block Fault (EEPROM, watchdog).
1	Analog to digital communications fault or calibration error.
5	User scaling error caused out of range values.
	I/O Fault Types for I/O Bus Fault Category
0	Genius IOC disabled all outputs on the bus because communications timed out between the PLC CPU and the Genius IOC.
1	Genius Bus fault. (No interrupt to the GBC for its turn on the bus within the time-out period.)
3	Genius IOC detected a conflict between its SBA and another device on the bus.
	I/O Fault Types for the Loss of Block Category
0	No reason specified.
1	Loss of A/D communications.
	I/O Fault Types for Excessive Faults Category
1	Genius IOC detected a high error count on the Genius I/O Bus and dropped off the bus for at least 1.5 seconds.
	I/O Fault Types for Configuration Mismatch Category
2	Model number mismatch detected by I/O scanner.
3	Non-existent I/O module detected by I/O scanner.
4	I/O type mismatch detected by I/O scanner.
8	Integral analog module detected; expansion analog module mismatch.
9	Broadcast Data Length mismatch.
A	A configured feature is not supported
	I/O Fault Types for GBC Software Exception Category
1	Incoming datagram queue is full.
2	The queue for Read/Write requests in the GBC is full. The requests may be from the Genius Bus or from COMMREQs.
3	The low priority mail queue from the GBC to the PLC is full. The response to the PLC was lost.
4	Genius background message requiring PLC action was received before PLC completed initialization. Message was ignored.
5	Genius Report Fault message was not processed because GBC software revision is too old.
6	Excessive usage of internal GBC memory. User should verify COMMREQ usage.

I/O Fault Description

The I/O fault description component provides a specific fault code when the I/O fault category is circuit fault (discrete circuit fault, analog circuit fault, low-level analog fault) or module fault. It is undefined for other faults, but is always set to zero. The next two tables list the possible fault descriptions.

Table B-19. I/O Fault Descriptions

Number	Description
	Fault Descriptions for Discrete Circuit Faults
01	Loss of user side power.
02	Short in user wiring (for Genius, current level greater than 20 Amps).
04	Sustained overcurrent (for Genius, current level greater than 2 Amps).
08	Very low or no current flow.
10	Switch temperature too high.
20	Genius smart switch failure.
83	Series 90-70 I/O individual point fault (also indicated for I/O Module Fault category).
84	Series 90-70 output fuse blown (also indicated for I/O Module Fault category).
	Fault Descriptions for Analog Circuit Faults
01	Input channel low alarm.
02	Input channel high alarm.
04	Input channel under range.
08	Input channel over range.
10	Open wire detected on input channel.
20	Output channel under range.
40	Output channel over range.
80	Expansion channel not responding.
80	Feedback error for Genius Current Source Analog Block.
	Fault Descriptions for Low-level Analog Circuit Faults *
20	Improper RTD connection or thermocouple reverse junction fault.
40	Cold junction sensor fault on thermocouple block or internal error in RTD block.
80	Input channel shorted (Genius RTD and Strain Gauge Blocks only).
	Fault Descriptions for Module Faults **
08	Genius EEPROM or NVRAM failure.
20	Genius calibration memory failure.
40	Genius shared RAM fault.
80	Genius internal circuit fault.
81	Watchdog timeout (discrete I/O modules only).
82	Aux fault on discrete I/O modules.
83	Series 90-70 I/O individual point fault (also indicated for CIRCUIT_FAULT category).
84	Series 90-70 output fuse blown (also indicated for CIRCUIT_FAULT category).
	Fault Descriptions for GENA Faults
80	Fault on a GENA analog or discrete point.
87	Fault on Remote I/O Scanner

* The following analog faults also apply when the low-level analog fault type is indicated: AI_LOW_ALARM, AI_HI_ALARM, AI_UNDER_RANGE, AI_OVER_RANGE, and OPEN_WIRE.

** These faults are reported under the HEADEND_FAULT fault type.

I/O Fault Specific Data

An I/O fault table entry may contain up to 21 bytes of I/O fault specific data. In general, this area contains additional information related to the fault. Not all entries contain I/O fault specific data. This section describes those that do. All but one of these faults uses five bytes of I/O fault specific data; the global memory fault uses the 21-byte entry. If a fault is not listed, it does not have I/O fault specific data.

Faults originated by the Genius Bus Controller always have at least one byte of I/O fault specific data. This byte is in addition to whatever other data might be present.

Circuit Fault

Circuit fault entries use one or two bytes of the fault specific data area. When the Genius Bus Controller reports the fault, the first byte is generated by the Genius Bus Controller. If the fault type is a GENA fault, the second byte contains the data that was reported from the GENA module in fault byte 2 of its "Report Fault" message. If the fault type is not a GENA fault, the second byte contains the circuit configuration and is encoded, as shown in table B-20.

Loss/Addition of Block

In the case of a Loss of Block or Addition of Block fault, four bytes of fault specific data are used. The first byte is encoded, as shown in table B-20. The second byte contains the block configuration and is encoded as shown in table B-20. The third byte specifies the number of input circuits possibly used, and the fourth byte specifies the number of output circuits possibly used.

Global Memory Fault

The Global Memory fault uses 10 data bytes. The first byte contains the Subnet group number. The other nine bytes contain the global variable name, formatted as a null-terminated string.

Forced/Unforced Circuit

Three bytes of fault specific data are present when a circuit force is added or removed (Forced Circuit or Unforced Circuit). The first byte contains a code from table B-20. The second byte contains the block configuration, and byte 3 contains the discrete/analog indication.

Loss of or Missing IOC

When the PLC CPU registers a Loss of or Missing IOC fault, it includes in the I/O fault specific data one of the codes in table B-20.

Other I/O Faults

In addition to those faults listed above, when the Genius Bus Controller reports one of the following faults, it includes one of the bytes in table B-20 in the I/O fault specific data.

- I/O Bus fault.
- Module fault.
- IOC software fault.
- Extra Genius block.
- IOC hardware fault.
- Excessive faults.
- GBC software exception.

Block Switch

When the fault category is Block Switch, five bytes of fault specific data are used. The first byte is encoded, as shown in table B-20. The second byte contains the block configuration and is encoded as shown in table B-20. The third byte specifies the number of input circuits possibly used, and the fourth byte specifies the number of output circuits possibly used. The last byte (byte 5) of fault specific data contains the rack/slot address of the Genius bus controller which controlled the bus that the block switched off. It is encoded with the slot number in the low four bits and the rack number in the high four bits.

Symbolic Fault Specific Data

The following table lists data that is required for four kinds of fault specific data:

- Block circuit configuration.
- Block usage indication.
- Discrete/analog indication.
- Loss of IOC error code.

Table B-20. I/O Fault Specific Data

Decimal Number	Hex Code	Description
		Circuit Configuration
	1	Circuit is an input.
	2	Circuit is an output.
	3	Circuit is an output with feedback.
		Block Configuration
	1	Block configured for inputs only.
	2	Block configured for outputs only.
	3	Block has both inputs and outputs.
		Discrete/Analog Indication
	1	Discrete block.
	2	Analog block.
		LOSS_OF_IOC Error Code
1	1	IOC failed to respond to a CPU request.
2	2	CPU and IOC lost synchronization.
3	3	CPU/IOC communications failed.
4	4	VME bus error.
5	5	VME bus error.
6	6	CPU/IOC communications failed.
7	7	CPU/IOC communications failed.
8	8	IOC failed to respond to a CPU request.
9	9	CPU/IOC communications failed.
10	A	VME bus error.
11	B	BME bus error.
12	C	CPU/IOC communications failed.
13	D	CPU/IOC communications failed.
14	E	CPU/IOC communications filed.
15	F	IOC failed to respond to a CPU request.
16	10	CPU/IOC communications failed.
17	11	CPU/IOC communications failed.
18	12	IOC failed to respond to a CPU request.
19	13	CPU/IOC communications failed.
20	14	CPU/IOC communications failed.
21	15	Internal I/O scanner fault detected.
22	16	IOC failed to respond to a CPU request.
23	17	IOC failed to respond to a CPU request.
24	18	IOC failed to respond to a CPU request.
25	19	IOC failed to respond to a CPU request.
26	1A	IOC failed to respond to a CPU request.
27	1B	CPU/IOC communications failed.
28	1C	VME bus error occurred while reading input data.
29	1D	VME bus error occurred while reading input diagnostics.
30	1E	CPU/IOC communications failed.
31	1F	VME bus error occurred while writing output data to IOC.
32	20	CPU/IOC communications failed.
33	21	CPU/IOC communications failed.
34	22	CPU/IOC communications failed.
35	23	VME bus error.
36	24	VME bus error.
37	25	Unable to read data from IOC for redundant I/O blocks.
38	26	Unable to write data to IOC for redundant I/O blocks.
39	27	IOC does not support configured I/O redundancy.
40	28	IOC failed to respond to a CPU request.
41	29	I/O scanner detected too many IOCs in the system.
42	2A	I/O scanner detected too many IOCs in the system.
43	2B	VME bus error.
44	2C	VME bus error.

Fault Actions for Specific Faults

Forced/unforced circuit faults are reported as informational faults. All others are diagnostic or fatal.

The model number mismatch, I/O type mismatch and non-existent I/O module faults are reported in the PLC fault table under the System Configuration Mismatch group. They are not reported in the I/O fault table.

I/O Fault Time Stamp

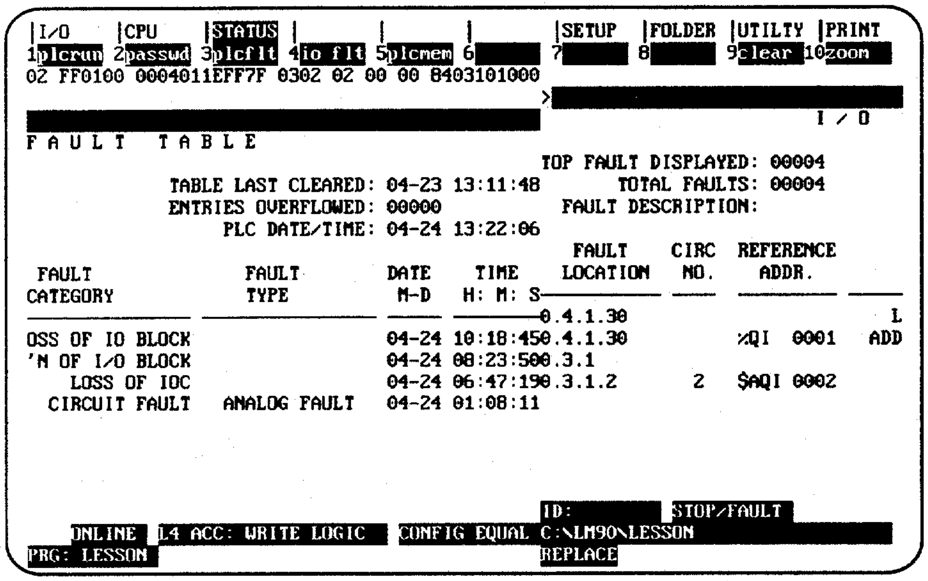
The six-byte time stamp is the value of the system clock when the fault was recorded by the PLC CPU. Values are coded in BCD format.

Table B-21. I/O Fault Time Stamp

Byte Number	Description
1	Seconds.
2	Minutes.
3	Hours.
4	Day of the month.
5	Month.
6	Year.

I/O Fault Table Examples

The following sample I/O fault table displays the fault specific information for the first fault on the line directly beneath the softkey menus.



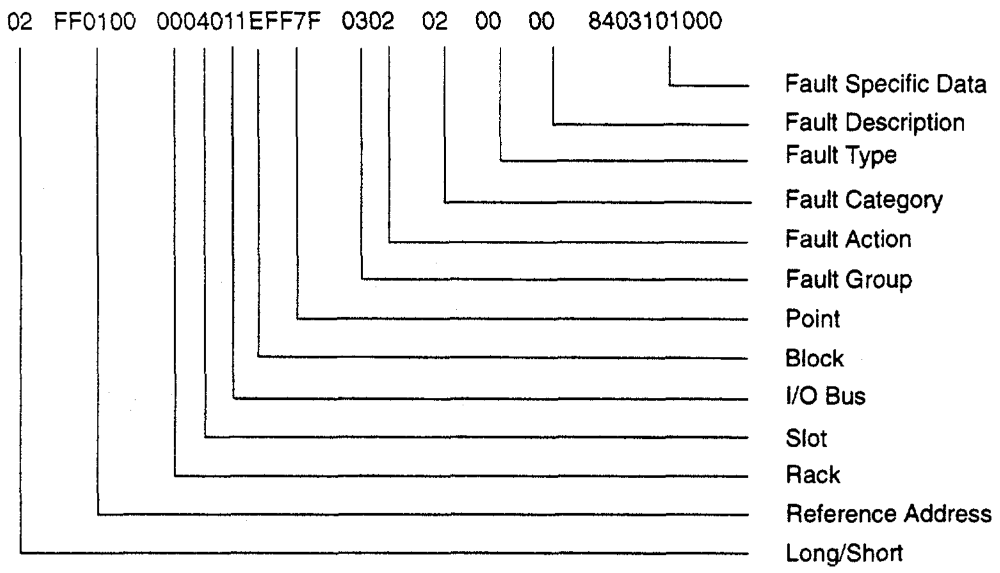
```
|I/O     |CPU     |STATUS |       |       |        |SETUP   |FOLDER  |UTILTY  |PRINT
1plcrun 2passwd 3plcflt 4io flt 5plcmem 6        7        8        9clear  10zoom
02 FF0100 0004011EFF7F 0302 02 00 00 8403101000
                                                    >
                                                                          I / O
F A U L T   T A B L E
                                                    TOP FAULT DISPLAYED: 00004
              TABLE LAST CLEARED: 04-23 13:11:48        TOTAL FAULTS: 00004
              ENTRIES OVERFLOWED: 00000           FAULT DESCRIPTION:
                   PLC DATE/TIME: 04-24 13:22:06
                                                     FAULT    CIRC  REFERENCE
 FAULT              FAULT          DATE    TIME     LOCATION   NO.    ADDR.
CATEGORY            TYPE           M-D    H: M: S────────────  ───   ─────────  ─────
───────────────  ──────────────  ─────  ──────────0.4.1.30                          L
OSS OF IO BLOCK                    04-24 10:18:450.4.1.30             %QI  0001   ADD
'N OF I/O BLOCK                    04-24 08:23:500.3.1
   LOSS OF IOC                     04-24 06:47:190.3.1.2        2     $AQI 0002
 CIRCUIT FAULT     ANALOG FAULT    04-24 01:08:11

                                                    ID:          STOP/FAULT
    ONLINE   L4 ACC: WRITE LOGIC   CONFIG EQUAL  C:\LM90\LESSON
PRG: LESSON                                         REPLACE
```

Loss of I/O Block Example

The loss of I/O Block fault is described below. (All data is in hexadecimal.)

Field	Value	Description
Long/Short	02	This fault contains up to 5 bytes of fault specific data.
Reference Address	FF0100	FF indicates that the reference address is not given. The fault applies to the entire block.
Rack	00	Main rack (rack 0).
Slot	04	Slot 4. In this configuration, slot 4 contains a Genius Bus Controller.
Bus	01	Bus 1 on the bus controller. When this byte is significant, the single channel bus controller always shows this field as a 1.
Block	1E	1E is 30 decimal. This fault was logged for the Genius device at serial bus address 30.
Point	FF7F	FF in the first (low byte) indicates that the point is not meaningful for this fault entry.
Fault Group	03	Loss of, or missing, I/O module fault.
Fault Action	02	Diagnostic fault.
Fault Category	02	Loss of block.
Fault Type	00	00 fault type in the Loss of Block category indicates no specific fault was given.
Fault Description	00	No fault description information is given.
Fault Specific Data	84	GBC reported a lost device. Only 1 byte of fault specific data is significant.

The Genius Bus Controller in slot 4 determined that the device at serial bus address 30 on bus 1 failed to send data in three consecutive bus scans. The bus controller then marked the device as lost and logged a fault in the I/O fault table. The fault type and fault description are not meaningful on this fault. The fault specific data contains a byte from the GBC echoing the fault.

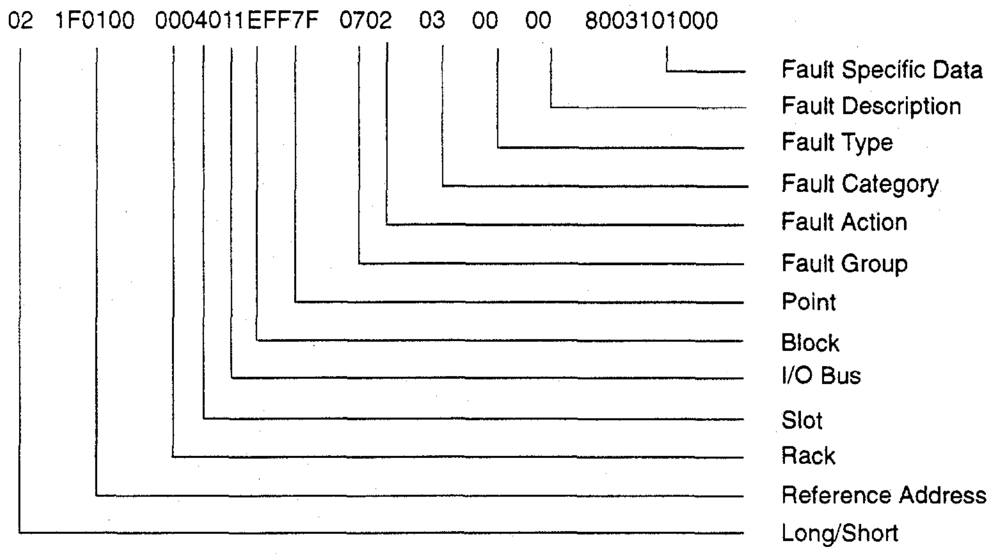


Addition of I/O Block Example

The Addition of I/O Block fault is described below. (All data is in hexadecimal.)

Field	Value	Description
Long/Short	02	This fault contains up to 5 bytes of fault specific data.
Reference Address	1F0100	1F (31 decimal) indicates that the block has a discrete grouped reference address. Logicmaster 90-70 software shows this by displaying a %QI in the reference address column. 0100 indicates this is address 0001 (hexadecimal data is displayed low byte first, then high byte) in the discrete grouped address space.
Rack	00	Main rack (rack 0).
Slot	04	Slot 4. In this configuration, slot 4 contains a Genius Bus Controller.
Bus	01	Bus 1 on the bus controller. When this byte is significant, the single channel bus controller always shows this field as a 1.
Block	1E	1E is 30 decimal. This fault was logged for the Genius device at serial bus address 30.
Point	FF7F	FF in the first (low byte) indicates that the point is not meaningful for this fault entry.
Fault Group	07	Addition of, or extra, I/O module fault.
Fault Action	02	Diagnostic fault.
Fault Category	03	Fault category 03 is an addition of block.
Fault Type	00	No fault type data occurs for an Addition of Block fault. The 00 is meaningless.
Fault Description	00	No fault description information is given.
Fault Specific Data	80	GBC reported an added device. Only 1 byte of fault specific data is significant.

The bus controller in slot 4 in the main rack received data from the device at serial bus address 30, indicating that the device was again on the bus. From the configuration table stored from Logicmaster 90-70 software to the PLC CPU, the CPU determined that the first point on the device was %QI0001. The fault type and fault description are not meaningful on this fault. The fault specific data contains a byte from the GBC echoing the fault.

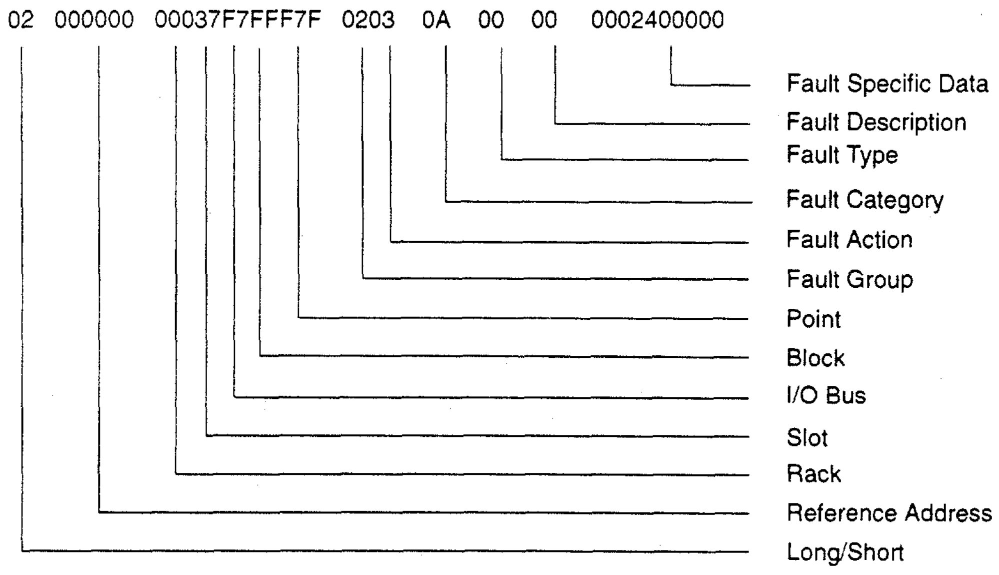


Loss of IOC

The Loss of IOC fault is explained below. (All values are in hexadecimal.)

Field	Value	Description
Long/Short	02	This fault contains up to 5 bytes of fault specific data.
Reference Address	000000	All zeros in this field indicates that the reference address is not meaningful for this fault.
Rack	00	Main rack (rack 0).
Slot	03	Slot 3. In this configuration, slot 3 contains a Genius Bus Controller.
Bus	7F	Since this is a single channel GBC, the bus number is not needed. 7F indicates that the bus number is not significant.
Block	7F	A block number of 7F hex indicates that the block is not meaningful for this fault.
Point	FF7F	FF in the first (low byte) indicates that the point is not meaningful for this fault entry.
Fault Group	02	Loss of, or missing, IOC fault.
Fault Action	03	Fatal fault.
Fault Category	0A	Fault Category 0A (10 decimal) is a Loss of, or Missing, IOC fault.
Fault Type	00	No fault type data occurs for a Loss of, or Missing, IOC fault.
Fault Description	00	No fault description information is given.
Fault Specific Data	00	There is no entry for a zero in fault specific data, so there is no additional information available on this instance of LOSS of IOC.

The PLC CPU detected a loss of, or missing, IOC (I/O Controller) and logged this fault. The fatal action indicates that the PLC CPU will not transition to RUN mode until the fault is cleared. Fault type and fault description are not meaningful for this fault; fault specific data may be meaningful.

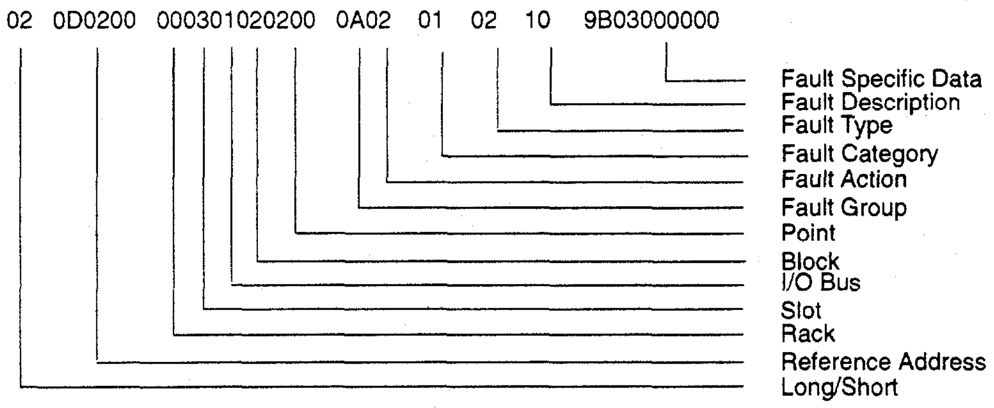


Circuit Fault

The Circuit Fault is explained below. (All values are in hexadecimal.)

Field	Value	Description
Long/Short	02	This fault contains up to 5 bytes of fault specific data.
Reference Address	0D0200	0D (13 decimal) indicates that the block has an analog grouped reference address. Logicmaster 90-70 software shows this by displaying a %AQI in the reference address column. 0200 indicates this is address 0002 (hexadecimal data is displayed low byte first, then high byte) in the analog grouped address space.
Rack	00	Main rack (rack 0).
Slot	03	Slot 3. In this configuration, slot 3 contains a Genius Bus Controller.
Bus	01	Bus 1 on the bus controller. When this byte is significant, the single channel bus controller always shows this field as a 1.
Block	02	This fault was logged for the Genius device at serial bus address 02.
Point	0200	This fault was logged for point 2 on the device at serial bus address 02.
Fault Group	0A	0A hexadecimal is 10 decimal. I/O Module fault.
Fault Action	02	Diagnostic fault.
Fault Category	01	Fault Category 01 is a circuit fault.
Fault Type	02	Fault Type 02 under the Circuit Fault category is a circuit fault on an analog I/O channel.
Fault Description	10	A fault description of 10 hex under the analog circuit fault description is open wire.
Fault Specific Data	9B03	The GBC reported a circuit fault. Only the 9B byte is significant.

The Genius Bus Controller in slot 3 in the main rack received a circuit fault "Report Fault" message from the analog block located at serial bus address 2. The block reported an Open Wire fault on point 2. From the configuration tables stored from Logicmaster 90-70 software, the PLC CPU determined that the analog block at serial bus address 02 was mapped to both the %AI and %AQ user references and that point 2 corresponded to location 0002.



Appendix C Instruction Mnemonics

In program display/edit mode, you can quickly enter or search for a program instruction by typing the ampersand (&) character followed by the instruction's mnemonic. For some instructions, you can also specify a reference address or nickname, a label, or a location reference address.

This appendix lists the mnemonics of the programming instructions for Logicmaster 90-70 programming software. At any time during programming, you can display a help screen with these mnemonics by pressing the **ALT** and **I** keys.

Table C-1. Contacts, Coils, Links, Timers, and Counters

Instruction	Mnemonic									
	All	INT	UINT	DINT	BIT	BYTE	WORD	DWORD	REAL	MIXED
Contacts										
Any Contact	&CON	&CON								
-| |-	&NOCON	&NOCON								
-|/|-	&NCCON	&NCCON								
-|↑|-	&PCON	&PCON								
-|↓|-	&NCON	&NCON								
-[FAULT]-	&FA	&FA								
-[NOFLT]-	&NOF	&NOF								
-[HIALR]-	&HI	&HI								
-[LOALR]-	&LOA	&LOA								
<+>---	&CONC	&CONC								
Coils										
Any Coil	&COI	&COI								
-( )-	&NOCOI	&NOCOI								
-(/)-	&NCCOI	&NCCOI								
-(↑)-	&PCOI	&PCOI								
-(↓)-	&NCOI	&NCOI								
-(S)-	&SL	&SL								
-(R)-	&RL	&RL								
-(SM)-	&SM	&SM								
-(RM)-	&RM	&RM								
-(M)-	&NOM	&NOM								
-(/M)-	&NCM	&NCM								
---<+>	&COILC	&COILC								
Links										
Horizontal	&HO	&HO								
Vertical	&VE	&VE								
Timers										
ONDTR	&ON	&ON								
OFDT	&OF	&OF								
TMR	&TM	&TM								
Counters										
UPCTR	&UP	&UP								
DNCTR	&DN	&DN								

Table C-2. Math, Relational, and Bit Operations

Instruction	Mnemonic									
	All	INT	UINT	DINT	BIT	BYTE	WORD	DWORD	REAL	MIXED
Math										
ADD	&AD	&AD_I	&AD_UI	&AD_DI					&AD_R	
SUB	&SUB	&SUB_I	&SUB_UI	&SUB_DI					&AD_R	
MUL	&MUL	&MUL_I	&MUL_UI	&MUL_DI					&MUL_R	&MUL_M
DIV	&DIV	&DIV_I	&DIV_UI	&DIV_DI					&DIV_R	&DIV_M
MOD	&MOD	&MOD_I	&MOD_UI	&MOD_DI						
SQRT	&SQ	&SQ_I		&SQ_DI					&SQ_R	
ABS	&ABS	&ABS_I		&ABS_DI					&ABS_R	
SIN	&SIN	&SIN								
COS	&COS	&COS								
TAN	&TAN	&TAN								
IASIN	&ASIN	&ASIN								
ACOS	&ACOS	&ACOS								
ATAN	&ATAN	&ATAN								
LOG	&LOG	&LOG								
LN	&LN	&LN								
EXP	&EXP	&EXP								
EXPT	&EXPT	&EXPT								
DEG	&DEG	&DEG								
RAD	&RAD	&RAD								
Relational										
EQ	&EQ	&EQ_I	&EQ_UI	&EQ_DI					&EQ_R	
NE	&NE	&NE_I	&NE_UI	&NE_DI					&NE_R	
GT	&GT	&GT_I	&GT_UI	&GT_DI					&GT_R	
GE	&GE	&GE_I	&GE_UI	&GE_DI					&GE_R	
LT	&LT	&LT_I	&LT_UI	&LT_DI					&LT_R	
LE	&LE	&LE_I	&LE_UI	&LE_DI					&LE_R	
CMP	&CM	&CM_I	&CM_UI	&CM_DI					&CM_R	
Bit Operation										
AND	&AN						&AN_W	&AN_DW		
OR	&OR						&OR_W	&OR_DW		
XOR	&XO						&XO_W	&XO_DW		
NOT	&NOT						&NOT_W	&NOT_DW		
SHL	&SHL						&SHL_W	&SHL_DW		
SHR	&SHR						&SHR_W	&SHR_DW		
ROL	&ROL						&ROL_W	&ROL_DW		
ROR	&ROR						&ROR_W	&ROR_DW		
BTST	&BT						&BT_W	&BT_DW		
BSET	&BS						&BS_W	&BS_DW		
BLCR	&BCL						&BCL_W	&BCL_DW		
BPOS	&BP						&BP_W	&BP_DW		
MCMP	&MCM						&MCM_W	&MCM_DW		

Table C-3. Data Move and Data Table Operations

Instruction	Mnemonic									
	All	INT	UINT	DINT	BIT	BYTE	WORD	DWORD	REAL	MIXED
Data Move										
MOVE	&MOV	&MOV_I	&MOV_UI	&MOV_DI	&MOV_BI		&MOV_W	&MOV_DW	&MOV_R	
BLKMOV	&BLKM	&BLKM_I	&BLKM_UI	&BLKM_DI			&BLKM_W	&BLKM_DW	&BLKM_R	
BLKCLR	&BLKC									
SHFR	&SHF				&SHF_BI		&SHF_W	&SHF_DW		
BITSEQ	&BI									
SWAP	&SW						&SW_W	&SW_DW		
COMMREQ	&COMMR									
VMERD	&VMERD					&VMERD_BY	&VMERD_W			
VMEWRT	&VMEW					&VMEWR_BY	&VMERD_W			
VMERMW	&VMERM					&VMERM_BY	&VMERM_W			
VMETST	&VMET					&VMET_BY	&VMET_W			
VME_CFG_RD	&CFGRD									
VME_CFG_WRT	&CFGWRT									
DATA_INIT	&DINI									
DATA_INIT_COMM	&DCO									
DATA_INIT_ASCII	&DA									
Data Table										
TBLRD	&TBLR	&TBLR_I	&TBLR_UI	&TBLR_DI			&TBLR_W	&TBLR_DW		
TBLWR	&TBLW	&TBLW_I	&TBLW_UI	&TBLW_DI			&TBLW_W	&TBLW_DW		
LIFORD	&LIFOR	&LIFOR_I	&LIFOR_UI	&LIFOR_DI			&LIFOR_W	&LIFOR_DW		
LIFOWRT	&LIFOW	&LIFOW_I	&LIFOW_UI	&LIFOW_DI			&LIFOW_W	&LIFOW_DW		
FIFORD	&FIFOR	&FIFOR_I	&FIFOR_UI	&FIFOR_DI			&FIFOR_W	&FIFOR_DW		
FIFOWRT	&FIFOW	&FIFOW_I	&FIFOW_UI	&FIFOW_DI			&FIFOW_W	&FIFOW_DW		
SORT	&SORT	&SORT_I	&SORT_UI				&SORT_W			
ARRAY_MOVE	&AR	&AR_I	&AR_UI	&AR_DI	&AR_BI	&AR_BY	&AR_W	&AR_DW		
SRCH_EQ	&SRCHE	&SRCHE_I	&SRCHE_UI	&SRCHE_DI		&SRCHE_BY	&SRCH_W	&SRCH_DW		
SRCH_NE	&SRCHN	&SRCHN_I	&SRCHN_UI	&SRCHN_DI		&SRCHN_BY	&SRCHN_W	&SRCHN_DW		
SRCH_GT	&SRCHGT	&SRCHGT_I	&SRCHGT_UI	&SRCHGT_DI		&SRCHGT_BY	&SRCHGT_W	&SRCHGT_DW		
SRCH_GE	&SRCHGE	&SRCHGE_I	&SRCHGE_UI	&SRCHGE_DI		&SRCHGE_BY	&SRCHGE_W	&SRCHGE_DW		
SRCH_LT	&SRCHLT	&SRCHLT_I	&SRCHLT_UI	&SRCHLT_DI		&SRCHLT_BY	&SRCHLT_W	&SRCHLT_DW		
SRCH_LE	&SRCHLE	&SRCHLE_I	&SRCHLE_UI	&SRCHLE_DI		&SRCHLE_BY	&SRCHLE_W	&SRCHLE_DW		

Table C-4. Conversion and Control Operations

Instruction	Mnemonic									
	All	INT	UINT	DINT	BIT	BYTE	WORD	DWORD	REAL	MIXED
Conversion										
to BCD-4	&BCD4	&BCD4	&BCD4_UI							
to BCD-8				&BCD8						
to UINT	&UI	&UI		&UI_DI					&UI_R	
to INT	&I	&I		&I_DI					&I_R	
to DINT	&DIN	&DIN	&DIN_UI						&DIN_R	
to REAL	&R	&R	&R_UI	&R_DI						
BCD-4 to UINT	&UI_BCD4									
BCD-4 to INT	&I_BCD4									
BCD-8 to DINT	&DI_BCD8									
BCD-4 to REAL	&R_BCD4									
BCD-8 to REAL	&R_BCD8									
TRUN	&TRINT	&TRINT		&TRDINT						
Control										
CALL	&CA									
DOIO	&DO									
SUSIO	&SUS									
MCR	&MCR									
ENDMCR	&ENDMCR									
JUMP	&JUMP									
LABEL	&LABEL									
COMMENT	&COMME									
SVCREQ	&SV									
PIDISA	&PIDIS									
PIDIND	&PIDIN									
FOR	&FOR									
END_FOR	&ENDFOR									
EXIT	&EXIT									

Appendix D Memory Allocation

Use this worksheet to determine the total number of bytes of memory used by %R, %AI, %AQ, %P, %L, and associated point faults, if used. Then, add all the totals together. To determine the amount of memory left for the user program, subtract the grand total from the number of bytes for the memory size being used.

Table D-1. Series 90-70 PLC Memory Allocation Worksheet

%R allocated = ____________ # of % R used x 2 Bytes = (# of %R used is in 1K increments based on configuration.)	____________
Note: When %P used is in the range 1 to 128, the %P allocated = 160. Otherwise, compute the number of %P allocated. %P Allocated = {[(# of %P used) – 128] / 32 rounded up to whole integer} * 32 + 160; %P Block Memory Used = (%P allocated) * 2 + 79 Bytes =	Totals ____________
Note: When %L used is in the range 1 to 64, the %L allocated = 96. Otherwise, compute the number of %L allocated. %L Allocated = {[(# of %L used) – 64] / 32 rounded up to whole integer} * 32 + 96; %L Block Memory Used = (%L allocated) * 2 + 79 Bytes =	____________

Note

%P or %L allocation for a program block cannot be changed while Logicmaster 90-70 software is online, unless no %P or %L is currently used in that block. To change the current %P or %L allocation, Logicmaster 90-70 software must be offline. The resulting program must then be stored to the PLC in **STOP** mode.

D-1. Series 90-70 PLC Memory Allocation Worksheet (cont'd)

%AI Allocated (If point fault **DISABLED**) = # %AI used x 2 (Bytes)	+ =	________
<u>or</u>		
%AI Allocated (If point fault **ENABLED**) = # %AI used x 3 (Bytes)	+ =	________
%AQ Allocated (If point fault **DISABLED**) = # %AQ used x 2 (Bytes)	+ =	________
<u>or</u>		
%AQ Allocated (If point fault **ENABLED**) = # %AQ used x 3 (Bytes)	+ =	________
If point fault is **ENABLED**, add: .... 128 bytes for 731/732 CPU <u>or</u> add: .... 512 bytes for 771/772 CPU <u>or</u> add: .... 3072 bytes for 781/782 CPU <u>or</u> add: .... 3072 bytes for 914–924 CPUs	+ =	________
GRAND Total (Bytes)	=	________

CPU Memory Total Bytes =	....	32K	=	327687	
OR	=	64K	=	65536	
OR	=	128K	=	131072	
OR	=	256K	=	262	
OR	=	512K	=	524288	= ________
Subtract Grand Total (From Above)					− ________
User Program Memory Available (In Bytes)					= ________

Note

Although 915 and 925 CPUs have a total of one megabyte (1024 KB), both CPUs have the same 512 KB limit for a single user program as other CPUs. The additional 512 kilobytes can be used for standalone C programs.

Appendix E Key Functions

The following table lists the keyboard functions that are active in the Logicmaster 90-70 software environment. This information may also be displayed on the programmer screen by pressing ALT-K to access key help.

Key Sequence	Description	Key Sequence	Description
Keys Available throughout the Software Package			
ALT-A	Abort.	CTRL-Break	Exit package.
ALT-C	Clear field.	Esc	Zoom out.
ALT-M	Change programmer mode.	CTRL-Home	Previous command line contents.
ALT-R	Change PLC **RUN/STOP** state.	CTRL-End	Next command line contents.
ALT-E	Toggle status area.	CTRL-←	Cursor left within the field.
ALT-L	List directory files.	CTRL-→	Cursor right within the field.
ALT-P	Print screen.	CTRL-D	Decrement reference address.
ALT-H	Help.	CTRL-U	Increment reference address.
ALT-K	Key help.	Tab	Change/increment field contents.
ALT-I	Instruction mnemonic help.	Shift-Tab	Change/decrement field contents.
ALT-T	Start **TEACH** mode.	Enter	Accept field contents.
ALT-Q	Stop **TEACH** mode.	CTRL-E	Display last system error.
ALT-Z	Pause **TEACH** playback.	F12 (or Keypad −)	Toggle discrete reference.
ALT-n	Playback file n (n = 0 thru 9).	F11 (or Keypad *)	Override discrete reference.
Keys Available in the Program Editor Only			
ALT-B	Toggle text editor bell.	Keypad +	Accept rung.
ALT-D	Delete rung element./Delete rung.	Enter	Accept rung.
ALT-S	Store block to PLC and disk.	CTRL-PgUp	Previous rung.
ALT-X	Display zoom level & block state.	CTRL-PgDn	Next rung.
ALT-V	Variable table window.	~	Horizontal link.
ALT-W	Display PSB parameter table.	|	Vertical link.
ALT-F2	Go to operand reference table.	Tab	Go to next operand field.
Special Keys			
ALT-G	Single sweep debug. Available only in program editor, reference table editor, and status fault tables.		
ALT-N	Toggle display options. Available only in program editor and reference table editor.		
ALT-O	Password override. Available only on Password screen in configuration software.		
ALT-U	Update disk.		

The Help card on the next page contains a listing of the key help and also the instruction mnemonics help text for Logicmaster 90 software. This card is printed in triplicate and is perforated for easier removal from the manual.

Logicmaster 90-70 Instruction Mnemonics Help (ALT-I)

Instruction	Mnemonic	Instruction	Mnemonic
Any contact	&CON	Bit Rotate Left	&ROL
Normally Open	&NOCON	Bit Rotate Right	&ROR
Normally Closed	&NCCON	Bit Test	&BT
Positive Transition	&PCON	Bit Set	&BS
Negative Transition	&NCON	Bit Clear	&BCL
Fault	&FA	Bit position	&BP
No Fault	&NOF	Masked Compare	&MCM
High Alarm	&HI	Mov	&MOV
Low Alarm	&LOA	Block Move	&BLKM
Continuation	&CONC	Block Clear	&BLKC
Any Coil	&COI	Shift Register	&SHF
Normally Open	&NOCOI	Bit Sequencer	&BI
Negated	&NCCOI	Swap Bytes	&SW
Positive Transition	&PCOI	Comm Request	&COMMR
Negative Transition	&NCOI	VME Read	&VMERD
SET	&SL	VME Write	&VMEW
RESET	&RL	VME Read/Modify/Write	&VMERM
Retentive SET	&SM	VME Test	&VMET
Retentive RESET	&RM	VME Config Read	&CFGRD
Retentive	&NOM	VME Config Write	&CFGWRT
Negated Retentive	&NCM	Data Initialize Type	&DINI
Continuation	&COILC	Data Init COMMREQ	&DCO
Horizontal Link	&HO	Data Initialize ASCII	&DA
Vertical Link	&VE	Table Read	&TBLR
On Delay Timer	&ON	Table Write	&TBLW
Off Delay Timer	&OF	LIFO Read	&LIFOR
Elapsed Timer	&TM	LIFO Write	&LIFOW
Up Counter	&UP	FIFO Read	&FIFOR
Down Counter	&DN	FIFO Write	&FIFOW
Addition	&AD	Sort	&SORT
Subtraction	&SUB	Array Move	&AR
Multiplication	&MUL	Search Equal	&SRCHE
Division	&DIV	Search Not Equal	&SRCHN
Modulo Divide	&MOD	Search Greater Than	&SRCHGT
Square Root	&SQ	Search Greater/Equal	&SRCHGE
Absolute Value	&ABS	Search Less Than	&SRCHLT
Sine	&SIN	Search Less Than/Equal	&SRCHLE
Cosine	&COS	Convert to BCD-4	&BCD4
Tangent	&TAN	Convert to BCD-8	&BCD8
Inverse Sine	&ASIN	Convert to UINT	&UI
Inverse Cosine	&ACOS	Convert to INT	&I
Inverse Tangent	&ATAN	Convert to DINT	&DIN
Base 10 Logarithm	&LOG	Convert to REAL	&R
Natural Logarithm	&LN	Convert BCD-4 to UINT	&UI_BCD4
Power of e	&EXP	Convert BCD-4 to INT	&I_BCD4
Power of x	&EXPT	Convert BCD-8 to DINT	&DI_BCD8
Convert to Degrees	&DEG	Convert BCD-4 to REAL	&R_BCD4
Convert to Radians	&RAD	Convert BCD-8 to REAL	&R_BCD8
Equal	&EQ	Truncate	&TRINT
Not Equal	&NE	Call a Program Block	&CA
Greater Than	&GT	Do I/O	&DO
Greater Than or Equal	&GE	Suspend I/O	&SUS
Less Than	&LT	Master Control Relay	&MCR
Less Than or Equal	&LE	End MCR	&ENDMCR
Compare	&CM	Jump	&JUMP
Bitwise AND	&AN	Label	&LABEL
Bitwise OR	&OR	Rung Explanation	&COMME
Bitwise Exclusive OR	&XO	Service Request	&SV
Bitwise NOT	&NOT	PID - ISA Algorithm	&PIDIS
Bit Shift Left	&SHL	PID - IND Algorithm	&PIDIN
Bit Shift Right	&SHR		

The following data types may be appended to an instruction by using an underscore (_) as a separator (e.g., &AD_I) Signed Integer (_I), Unsigned Integer (_UI), Double Precision Integer (_DI), Bit (_BI), Byte (_BY), Word (_W), Double Word (_DW), Floating Point (_R), and Mixed (_M)

Logicmaster 90-70 Instruction Mnemonics Help (ALT-I)

Instruction	Mnemonic	Instruction	Mnemonic
Any contact	&CON	Bit Rotate Left	&ROL
Normally Open	&NOCON	Bit Rotate Right	&ROR
Normally Closed	&NCCON	Bit Test	&BT
Positive Transition	&PCON	Bit Set	&BS
Negative Transition	&NCON	Bit Clear	&BCL
Fault	&FA	Bit position	&BP
No Fault	&NOF	Masked Compare	&MCM
High Alarm	&HI	Mov	&MOV
Low Alarm	&LOA	Block Move	&BLKM
Continuation	&CONC	Block Clear	&BLKC
Any Coil	&COI	Shift Register	&SHF
Normally Open	&NOCOI	Bit Sequencer	&BI
Negated	&NCCOI	Swap Bytes	&SW
Positive Transition	&PCOI	Comm Request	&COMMR
Negative Transition	&NCOI	VME Read	&VMERD
SET	&SL	VME Write	&VMEW
RESET	&RL	VME Read/Modify/Write	&VMERM
Retentive SET	&SM	VME Test	&VMET
Retentive RESET	&RM	VME Config Read	&CFGRD
Retentive	&NOM	VME Config Write	&CFGWRT
Negated Retentive	&NCM	Data Initialize Type	&DINI
Continuation	&COILC	Data Init COMMREQ	&DCO
Horizontal Link	&HO	Data Initialize ASCII	&DA
Vertical Link	&VE	Table Read	&TBLR
On Delay Timer	&ON	Table Write	&TBLW
Off Delay Timer	&OF	LIFO Read	&LIFOR
Elapsed Timer	&TM	LIFO Write	&LIFOW
Up Counter	&UP	FIFO Read	&FIFOR
Down Counter	&DN	FIFO Write	&FIFOW
Addition	&AD	Sort	&SORT
Subtraction	&SUB	Array Move	&AR
Multiplication	&MUL	Search Equal	&SRCHE
Division	&DIV	Search Not Equal	&SRCHN
Modulo Divide	&MOD	Search Greater Than	&SRCHGT
Square Root	&SQ	Search Greater/Equal	&SRCHGE
Absolute Value	&ABS	Search Less Than	&SRCHLT
Sine	&SIN	Search Less Than/Equal	&SRCHLE
Cosine	&COS	Convert to BCD 4	&BCD4
Tangent	&TAN	Convert to BCD 8	&BCD8
Inverse Sine	&ASIN	Convert to UINT	&UI
Inverse Cosine	&ACOS	Convert to INT	&I
Inverse Tangent	&ATAN	Convert to DINT	&DIN
Base 10 Logarithm	&LOG	Convert to REAL	&R
Natural Logarithm	&LN	Convert BCD 4 to UINT	&UI_BCD4
Power of e	&EXP	Convert BCD 4 to INT	&I_BCD4
Power of x	&EXPT	Convert BCD 8 to DINT	&DI_BCD8
Convert to Degrees	&DEG	Convert BCD 4 to REAL	&R_BCD4
Convert to Radians	&RAD	Convert BCD 8 to REAL	&R_BCD8
Equal	&EQ	Truncate	&TRINT
Not Equal	&NE	Call a Program Block	&CA
Greater Than	&GT	Do I/O	&DO
Greater Than or Equal	&GE	Suspend I/O	&SUS
Less Than	&LT	Master Control Relay	&MCR
Less Than or Equal	&LE	End MCR	&ENDMCR
Compare	&CM	Jump	&JUMP
Bitwise AND	&AN	Label	&LABEL
Bitwise OR	&OR	Rung Explanation	&COMME
Bitwise Exclusive OR	&XO	Service Request	&SV
Bitwise NOT	&NOT	PID - ISA Algorithm	&PIDIS
Bit Shift Left	&SHL	PID - IND Algorithm	&PIDIN
Bit Shift Right	&SHR		

The following data types may be appended to an instruction by using an underscore (_) as a separator (e.g., &AD_I) Signed Integer (_I), Unsigned Integer (_UI), Double Precision Integer (_DI), Bit (_BI), Byte (_BY), Word (_W), Double Word (_DW), Floating Point (_R), and Mixed (_M)

Logicmaster 90-70 Instruction Mnemonics Help (ALT-I)

Instruction	Mnemonic	Instruction	Mnemonic
Any contact	&CON	Bit Rotate Left	&ROL
Normally Open	&NOCON	Bit Rotate Right	&ROR
Normally Closed	&NCCON	Bit Test	&BT
Positive Transition	&PCON	Bit Set	&BS
Negative Transition	&NCON	Bit Clear	&BCL
Fault	&FA	Bit position	&BP
No Fault	&NOF	Masked Compare	&MCM
High Alarm	&HI	Mov	&MOV
Low Alarm	&LOA	Block Move	&BLKM
Continuation	&CONC	Block Clear	&BLKC
Any Coil	&COI	Shift Register	&SHF
Normally Open	&NOCOI	Bit Sequencer	&BI
Negated	&NCCOI	Swap Bytes	&SW
Positive Transition	&PCOI	Comm Request	&COMMR
Negative Transition	&NCOI	VME Read	&VMERD
SET	&SL	VME Write	&VMEW
RESET	&RL	VME Read/Modify/Write	&VMERM
Retentive SET	&SM	VME Test	&VMET
Retentive RESET	&RM	VME Config Read	&CFGRD
Retentive	&NOM	VME Config Write	&CFGWRT
Negated Retentive	&NCM	Data Initialize Type	&DINI
Continuation	&COILC	Data Init COMMREQ	&DCO
Horizontal Link	&HO	Data Initialize ASCII	&DA
Vertical Link	&VE	Table Read	&TBLR
On Delay Timer	&ON	Table Write	&TBLW
Off Delay Timer	&OF	LIFO Read	&LIFOR
Elapsed Timer	&TM	LIFO Write	&LIFOW
Up Counter	&UP	FIFO Read	&FIFOR
Down Counter	&DN	FIFO Write	&FIFOW
Addition	&AD	Sort	&SORT
Subtraction	&SUB	Array Move	&AR
Multiplication	&MUL	Search Equal	&SRCHE
Division	&DIV	Search Not Equal	&SRCHN
Modulo Divide	&MOD	Search Greater Than	&SRCHGT
Square Root	&SQ	Search Greater/Equal	&SRCHGE
Absolute Value	&ABS	Search Less Than	&SRCHLT
Sine	&SIN	Search Less Than/Equal	&SRCHLE
Cosine	&COS	Convert to BCD-4	&BCD4
Tangent	&TAN	Convert to BCD-8	&BCD8
Inverse Sine	&ASIN	Convert to UINT	&UI
Inverse Cosine	&ACOS	Convert to INT	&I
Inverse Tangent	&ATAN	Convert to DINT	&DIN
Base 10 Logarithm	&LOG	Convert to REAL	&R
Natural Logarithm	&LN	Convert BCD 4 to UINT	&UI_BCD4
Power of e	&EXP	Convert BCD-4 to INT	&I_BCD4
Power of x	&EXPT	Convert BCD-8 to DINT	&DI_BCD8
Convert to Degrees	&DEG	Convert BCD 4 to REAL	&R_BCD4
Convert to Radians	&RAD	Convert BCD-8 to REAL	&R_BCD8
Equal	&EQ	Truncate	&TRINT
Not Equal	&NE	Call a Program Block	&CA
Greater Than	&GT	Do I/O	&DO
Greater Than or Equal	&GE	Suspend I/O	&SUS
Less Than	&LT	Master Control Relay	&MCR
Less Than or Equal	&LE	End MCR	&ENDMCR
Compare	&CM	Jump	&JUMP
Bitwise AND	&AN	Label	&LABEL
Bitwise OR	&OR	Rung Explanation	&COMME
Bitwise Exclusive OR	&XO	Service Request	&SV
Bitwise NOT	&NOT	PID - ISA Algorithm	&PIDIS
Bit Shift Left	&SHL	PID - IND Algorithm	&PIDIN
Bit Shift Right	&SHR		

The following data types may be appended to an instruction by using an underscore (_) as a separator (e.g., &AD_I) Signed Integer (_I), Unsigned Integer (_Ui), Double Precision Integer (_DI), Bit (_BI), Byte (_BY), Word (_W), Double Word (_DW), Floating Point (_R), and Mixed (_M)

E

GE FANUC

Logicmaster Key Help (ALT-K) **GFJ-056B**

Key Sequence	Description
Keys Available throughout the Software Package	
ALT-A	Abort.
ALT-C	Clear field.
ALT-M	Change programmer mode.
ALT-R	Change PLC **RUN/STOP** state.
ALT-E	Toggle status area.
ALT-L	List directory files.
ALT-P	Print screen.
ALT-H	Help.
ALT-K	Key help.
ALT-I	Instruction mnemonic help.
ALT-T	Start **TEACH** mode.
ALT-Q	Stop **TEACH** mode.
ALT-Z	Pause **TEACH** playback.
ALT-n	Playback file n (n = 0 thru 9).
CTRL-Break	Exit package.
Esc	Zoom out.
CTRL-Home	Previous command line contents.
CTRL-End	Next command line contents.
CTRL-←	Cursor left within the field.
CTRL-→	Cursor right within the field.
CTRL-D	Decrement reference address.
CTRL-U	Increment reference address.
Tab	Change/increment field contents.
Shift-Tab	Change/decrement field contents.
Enter	Accept field contents.
CTRL-E	Display last system error.
F12 (or Keypad −)	Toggle discrete reference.
F11 (or Keypad *)	Override discrete reference.
Keys Available in the Program Editor Only	
ALT-B	Toggle text editor bell.
ALT-D	Delete rung element./Delete rung.
ALT-S	Store block to PLC and disk.
ALT-X	Display zoom level and block state.
ALT-V	Variable table window.
ALT-W	Display PSB parameter table.
ALT-F2	Go to operand reference table.
Keypad +	Accept rung.
Enter	Accept rung.
CTRL-PgUp	Previous rung.
CTRL-PgDn	Next rung.
-	Horizontal link.
|	Vertical link.
Tab	Go to next operand field.
Special Keys	
ALT-G	Single sweep debug. Available only in program editor, reference table editor, and status fault tables.
ALT-N	Toggle display options. Available only in program editor and reference table editor.
ALT-O	Password override. Available only on Password screen in the configuration software.
ALT-U	Update disk.

GE FANUC

Logicmaster Key Help (ALT-K) **GFJ-056B**

Key Sequence	Description
Keys Available throughout the Software Package	
ALT-A	Abort.
ALT-C	Clear field.
ALT-M	Change programmer mode.
ALT-R	Change PLC **RUN/STOP** state.
ALT-E	Toggle status area.
ALT-L	List directory files.
ALT-P	Print screen.
ALT-H	Help.
ALT-K	Key help.
ALT-I	Instruction mnemonic help.
ALT-T	Start **TEACH** mode.
ALT-Q	Stop **TEACH** mode.
ALT-Z	Pause **TEACH** playback.
ALT-n	Playback file n (n = 0 thru 9).
CTRL-Break	Exit package.
Esc	Zoom out.
CTRL-Home	Previous command line contents.
CTRL-End	Next command line contents.
CTRL-←	Cursor left within the field.
CTRL-→	Cursor right within the field.
CTRL-D	Decrement reference address.
CTRL-U	Increment reference address.
Tab	Change/increment field contents.
Shift-Tab	Change/decrement field contents.
Enter	Accept field contents.
CTRL-E	Display last system error.
F12 (or Keypad −)	Toggle discrete reference.
F11 (or Keypad *)	Override discrete reference.
Keys Available in the Program Editor Only	
ALT-B	Toggle text editor bell.
ALT-D	Delete rung element./Delete rung.
ALT-S	Store block to PLC and disk.
ALT-X	Display zoom level and block state.
ALT-V	Variable table window.
ALT-W	Display PSB parameter table.
ALT-F2	Go to operand reference table.
Keypad +	Accept rung.
Enter	Accept rung.
CTRL-PgUp	Previous rung.
CTRL-PgDn	Next rung.
-	Horizontal link.
|	Vertical link.
Tab	Go to next operand field.
Special Keys	
ALT-G	Single sweep debug. Available only in program editor, reference table editor, and status fault tables.
ALT-N	Toggle display options. Available only in program editor and reference table editor.
ALT-O	Password override. Available only on Password screen in the configuration software.
ALT-U	Update disk.

GE FANUC

Logicmaster Key Help (ALT-K) **GFJ-056B**

Key Sequence	Description
Keys Available throughout the Software Package	
ALT-A	Abort.
ALT-C	Clear field.
ALT-M	Change programmer mode.
ALT-R	Change PLC **RUN/STOP** state.
ALT-E	Toggle status area.
ALT-L	List directory files.
ALT-P	Print screen.
ALT-H	Help.
ALT-K	Key help.
ALT-I	Instruction mnemonic help.
ALT-T	Start **TEACH** mode.
ALT-Q	Stop **TEACH** mode.
ALT-Z	Pause **TEACH** playback.
ALT-n	Playback file n (n = 0 thru 9).
CTRL-Break	Exit package.
Esc	Zoom out.
CTRL-Home	Previous command line contents.
CTRL-End	Next command line contents.
CTRL-←	Cursor left within the field.
CTRL-→	Cursor right within the field.
CTRL-D	Decrement reference address.
CTRL-U	Increment reference address.
Tab	Change/increment field contents.
Shift-Tab	Change/decrement field contents.
Enter	Accept field contents.
CTRL-E	Display last system error.
F12 (or Keypad −)	Toggle discrete reference.
F11 (or Keypad *)	Override discrete reference.
Keys Available in the Program Editor Only	
ALT-B	Toggle text editor bell.
ALT-D	Delete rung element./Delete rung.
ALT-S	Store block to PLC and disk.
ALT-X	Display zoom level and block state.
ALT-V	Variable table window.
ALT-W	Display PSB parameter table.
ALT-F2	Go to operand reference table.
Keypad +	Accept rung.
Enter	Accept rung.
CTRL-PgUp	Previous rung.
CTRL-PgDn	Next rung.
-	Horizontal link.
|	Vertical link.
Tab	Go to next operand field.
Special Keys	
ALT-G	Single sweep debug. Available only in program editor, reference table editor, and status fault tables.
ALT-N	Toggle display options. Available only in program editor and reference table editor.
ALT-O	Password override. Available only on Password screen in the configuration software.
ALT-U	Update disk.

Appendix F Using Floating-Point Numbers

There are a few considerations you need to understand when using floating-point numbers. The first section discusses these general considerations. Refer to page F-5 and following for instructions on entering and displaying floating-point numbers.

Note

Use of floating-point numbers within a parameterized subroutine block (PSB) usually necessitates the use of NWORD parameters rather than WORD parameters. For information about when to use NWORDs within PSBs, refer to the "Formal Parameters within a Parameterized Subroutine Block" part of section 10, "Parameterized Subroutines," in chapter 3, "Program Editing," of the *Logicmaster 90-70 Programming Software User's Manual,* GFK-0263.

Floating-Point Numbers

Logicmaster 90 software provides the ability to edit, display, store, and retrieve numbers with real values. Some functions operate on floating-point numbers. However, in order to use floating-point numbers with Logicmaster 90-70 software, you must have a 732, 772, or 782 , 914 or higher CPU. Floating-point numbers are represented in decimal scientific notation, with a display of six significant digits.

Note

In this manual, the terms "floating-point" and "REAL" are used interchangeably to describe the floating-point number display/entry feature of Logicmaster software.

In Logicmaster 90 software, the following format is used. For numbers in the range 9999999999 to .0001, the display has no exponent and up to six or seven significant digits. For example:

Entered	Displayed	Description
.000123456789	+.0001234567	Ten digits, six or seven significant.
−12.345e-2	−.1234500	Seven digits, six or seven significant.
1234	+1234.000	Seven digits, six or seven significant.

Outside the range listed above, only six significant digits are displayed and the display has the form:

```
+1.23456E+12
 |||  |  | |
 |||  |  | +—— Exponent (signed power of 10)
 |||  |  |
 |||  |  +———— Exponent indicator and sign of exponent
 |||  |
 |||  +——————— Five less significant digits
 |||
 ||+——————————— Decimal point
 ||
 |+———————————— Most significant digit
 |
 +————————————— Sign of the entire number
```

Values of Floating-Point Numbers

Use the following table to calculate the value of a floating-point number from the binary number stored in two registers.

Exponent (e)	Mantissa (f)	Value of Floating Point Number
255	Non-zero	Not a valid number (NaN).
255	0	$-1^{s} * \infty$
$0 < e < 255$	Any value	$-1^{s} * 2^{e-127} * 1.f$
0	Non-zero	$-1^{s} * 2^{-126} * 0.f$
0	0	0

f = the mantissa. The mantissa is a binary fraction.
e = the exponent. The exponent is an integer E such that E+127 is the power of 2 by which the mantissa must be multiplied to yield the floating-point value.
s = the sign bit.
* = the multiplication operator.

For example, consider the floating-point number 12.5. The IEEE floating-point binary representation of the number is:

```
01000001 01001000 00000000 00000000
```

or 41480000 hex in hexadecimal form. The most significant bit (the sign bit) is zero (s=0). The next eight most significant bits are 10000010, or 130 decimal (e=130).

The mantissa is stored as a decimal binary number with the decimal point preceding the most significant of the 23 bits. Thus, the most significant bit in the mantissa is a multiple of 2^{-1}, the next most significant bit is a multiple of $2-^{2}$, and so on to the least significant bit, which is a multiple of 2^{-23}. The final 23 bits (the mantissa) are:

```
1001000 00000000 00000000
```

The value of the mantissa, then, is .5625 (that is, $2^{-1} + 2^{-4}$).

Since e > 0 and e < 255, we use the third formula in the table above:

$$\begin{aligned}\textbf{number} &= -1^{s} * 2^{e-127} * 1.f\\ &= -1^{0} * 2^{130-127} * 1.5625\\ &= 1 * 2^{3} * 1.5625\\ &= 8 * 1.5625\\ &= 12.5\end{aligned}$$

Thus, you can see that the above binary representation is correct.

The range of numbers that can be stored in this format is from ± 1.401298E−45 to ± 3.402823E+38 and the number zero.

Errors in Floating-Point Numbers and Operations

Overflow occurs when a number greater than 3.402823E+38 or less than −3.402823E+38 is generated by a REAL function. When this occurs, the ok output of the function is set OFF; and the result is set to positive infinity (for a number greater than 3.402823E+38) or negative infinity (for a number less than −3.402823E+38). You can determine where this occurs by testing the sense of the ok output.

POS_INF	= 7F800000h	– IEEE positive infinity representation in hex.
NEG_INF	= FF800000h	– IEEE negative infinity representation in hex.

If the infinities produced by overflow are used as operands to other REAL functions, they may cause an undefined result. This undefined result is referred to as an NaN (Not a Number). For example, the result of adding positive infinity to negative infinity is undefined. When the ADD_REAL function is invoked with positive infinity and negative infinity as its operands, it produces an NaN for its result.

Each REAL function capable of producing an NaN produces a specialized NaN which identifies the function.

NaN_ADD.	= 7F81FFFFh	– Real addition error value in hex.
NaN_SUB	= 7F81FFFFh	– Real subtraction error value in hex.
NaN_MUL	= 7F82FFFFh	– Real multiplication error value in hex.
NaN_DIV	= 7F83FFFFh	– Real division error value in hex.
NaN_SQRT	= 7F84FFFFh	– Real square root error value in hex.
NaN_LOG	= 7F85FFFFh	– Real logarithm error value in hex.
NaN_POW0	= 7F86FFFFh	– Real exponent error value in hex.
NaN_SIN	= 7F87FFFFh	– Real sine error value in hex.
NaN_COS	= 7F88FFFFh	– Real cosine error value in hex.
NaN_TAN	= 7F89FFFFh	– Real tangent error value in hex.
NaN_ASIN	= 7F8AFFFFh	– Real inverse sine error value in hex.
NaN_ACOS	= 7F8BFFFFh	– Real inverse cosine error value in hex.
NaN_BCD	= 7F8CFFFFh	– BCD-4 to real error.
REAL_INDEF	= FFC00000	– Real indefinite, divide 0 by 0 error.

When an NaN result is fed into another function, it passes through to the result. For example, if an NaN_ADD is the first operand to the SUB_REAL function, the result of the SUB_REAL is NaN_ADD. If both operands to a function are NaNs, the first operand will pass through. Because of this feature of propagating NaNs through functions, you can identify the function where the NaN originated.

Note

For NaN, the ok output is OFF (not energized).

Entering and Displaying Floating-Point Numbers

In the mantissa, up to six or seven significant digits of precision may be entered and stored; however, Logicmaster 90 software will display only the first six of these digits. The mantissa may be preceded by a positive or negative sign. If no sign is entered, the floating-point number is assumed to be positive.

If an exponent is entered, it must be preceded by the letter **E** or **e**, and the mantissa must contain a decimal point to avoid mistaking it for a hexadecimal number. The exponent may be preceded by a sign; but, if none is provided, it is assumed to be positive. If no exponent is entered, it is assumed to be zero. No spaces are allowed in a floating-point number.

To provide ease-of-use, several formats are accepted in both command-line and field data entry. These formats include an integer, a decimal number, or a decimal number followed by an exponent. These numbers are converted to a standard form for display once the user has entered the data and pressed the **Enter** key.

Examples of valid floating-point number entries and their normalized display are shown below.

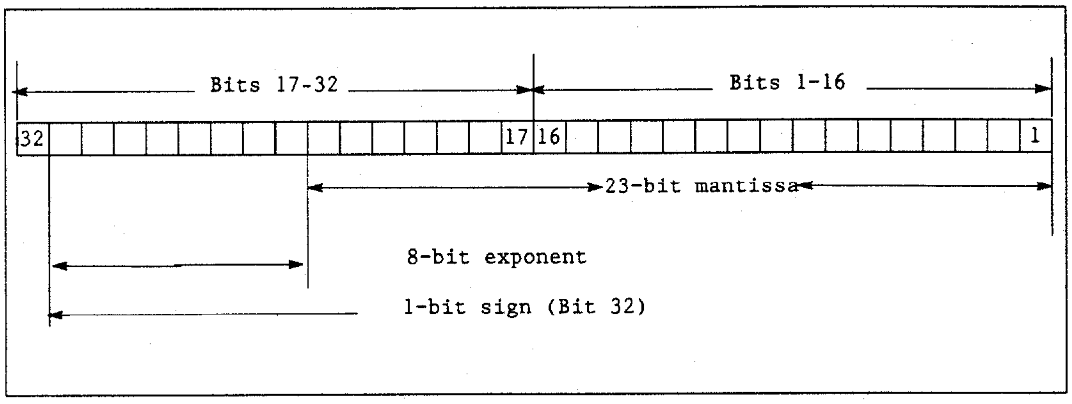
Entered	Displayed
250	+250,0000
+4	+4.000000
−2383019	−2383019.
34.	+34.00000
−.0036209	−.003620900
12.E+9	+1.20000E+10
−.0004E−11	−4.00000E−15
731.0388	+731.0388
99.20003e−29	+9.92000E−28

Examples of invalid floating-point number entries are shown below.

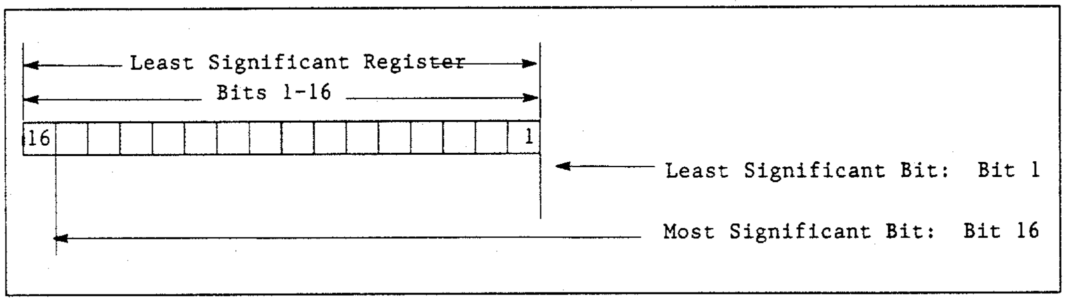
Invalid Entry	Explanation
−433E23	Missing decimal point.
10e-19	Missing decimal point.
10.e19	The mantissa cannot contain spaces between digits or characters. This is accepted as 10.e0, and an error message is displayed.
4.1e19	The exponent cannot contain spaces between digits or characters. This is accepted as 4.1e0, and an error message is displayed.

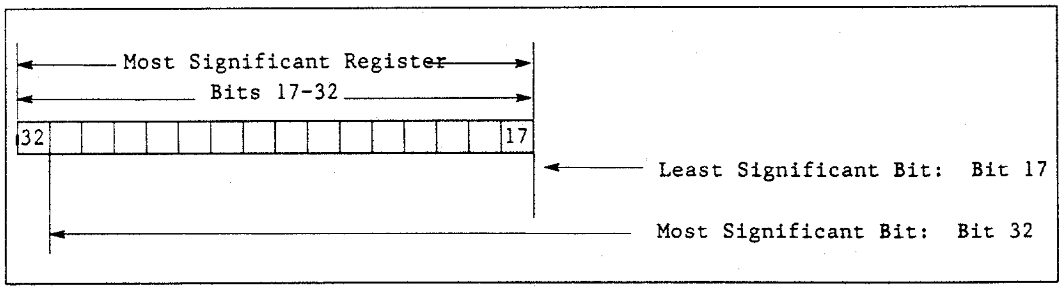
Internal Format of Floating-Point Numbers

Floating-point numbers are stored in single precision IEEE-standard format. This format requires 32 bits, which translates to two (adjacent) 16-bit PLC registers. The encoding of the bits is diagrammed below.



Register use by a single floating-point number is diagrammed below. In this diagram, if the floating-point number occupies registers R5 and R6, for example, then R5 is the least significant register and R6 is the most significant register.


Least Significant Register
Bits 1-16
16
1
Least Significant Bit: Bit 1
Most Significant Bit: Bit 16


Most Significant Register
Bits 17-32
32
17
Least Significant Bit: Bit 17
Most Significant Bit: Bit 32

Index

GE Fanuc Automation North America, Inc., Charlottesville, Virginia

GE Fanuc Automation

P.O. Box 8106
Charlottesville, VA 22906

GFK-0265G

FANUC

GE Fanuc Automation

Programmable Control Products

Logicmaster™ 90 Series 90™-30/20/Micro Programming Software

User's Manual

FANUC

GE Fanuc Automation

Programmable Control Products

Logicmaster™ 90 Series 90™-30/20/Micro Programming Software

User's Manual

GFK-0466G

March 1995

Warnings, Cautions, and Notes as Used in this Publication

Warning

Warning notices are used in this publication to emphasize that hazardous voltages, currents, temperatures, or other conditions that could cause personal injury exist in this equipment or may be associated with its use.

In situations where inattention could cause either personal injury or damage to equipment, a Warning notice is used.

Caution

Caution notices are used where equipment might be damaged if care is not taken.

Note

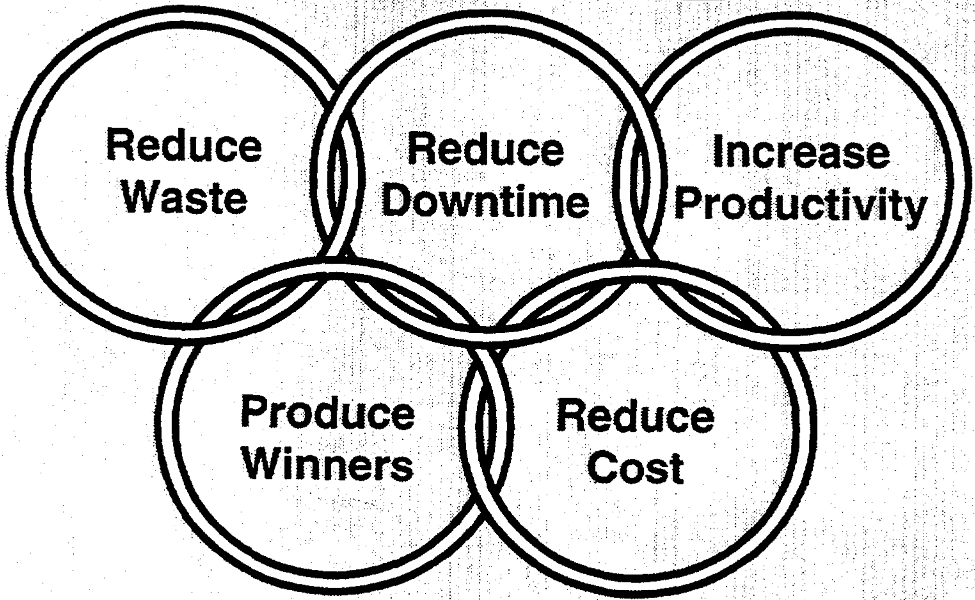
Notes merely call attention to information that is especially significant to understanding and operating the equipment.

Alarm Master	CIMSTAR	Helpmate	PROMACRO	Series Six
CIMPLICITY	GEnet	Logicmaster	Series One	Series 90
CIMPLICITY 90-ADS	Genius	Modelmaster	Series Three	VuMaster
CIMPLICITY PowerTRAC	Genius PowerTRAC	ProLoop	Series Five	Workmaster

GFA-142A

GE FANUC

World Champions!



Training Produces Champions!

In today's marketplace, downtime and engineering cost can erode your competitive edge.

In order to be a champion in any field, you must train. The business world is no different. Training improves morale, cuts costs and pays dividends. It is not only a smart investment, but a necessity if you want to be a world leader.

GE Fanuc Automation North America, Inc.

For more information on training with us.

PLC	1-800-828-5747	
CNC	1-800-828-5747	
	FAX	1-804-293-3900
CIMPLICITY	1-800-762-6498	
	FAX	1-518-464-4613

GE Fanuc Automation North America, Inc, Charlottesville, Virginia

Preface

This manual describes the features that are used to create ladder logic user programs for Series 90™-30 PLCs, Series 90™-20 PLCs, and Series 90™ Micro PLCs. These features are available with Release 5 of the Logicmaster™ 90-30/20/Micro programming software.

Revisions to This Manual

Changes made to this manual reflect the features of Release 5 (November 1994) of Logicmaster 90-30/20/Micro software for Series 90-30 PLCs, Series 90-20 PLCs, and Series 90 Micro PLCs. Additionally, corrections have been made where necessary. The following list describes the major revisions of this manual, as compared to the two previous versions (GFK-0466D and GFK-0466E):

- This manual includes software-related information about the recently released Model 351 CPU, such as folder (refer to page 7-4) and storage considerations (refer to page 8-6). For more information about the 351 CPU, refer to the *Series 90™-30 Installation Manual* (GFK-0356H) and the IPI that comes with the CPU.
- There is a new Series 90 Micro PLC, Model IC693UDD005, discussed briefly in the Micro PLC configuration section. This Micro PLC has 28 DC inputs and 28 relay outputs. For more information, refer to the *Series 90 Micro PLC User's Manual* (GFK-1065).
- There is a new TCP/IP Ethernet module. For configuration information, refer to page 10-60 and following and to the *TCP/IP Ethernet Communications for the Series 90™-30 PLC* manual (GFK-1084).
- You now have the ability to import Variable Declarations into the Variable Declarations Table using Comma Separated Variable (CSV) format and to export Variable Declarations from the Variable Declarations Table with Logicmaster placing the variables into Comma Separated Variable (CSV) format. This allows you to create and edit your Variable Declarations using a spreadsheet program. For information on how to use Logicmaster to import and export variables in CSV format, refer to page 3-41 and following. Refer to appendix I, "Variable Declaration Table Import/Export Using Comma Separated Variable (CSV) Format" for information on the format itself.
- In our effort to improve the quality of Logicmaster documentation, there are clarifications and corrections in several places within this manual. In addition to minor clarifications, we reorganized and improved the section on Communication Requests in the *Series 90™-30/20/Micro Programmable Controllers Reference Manual* (GFK-0467F).

Content of This Manual

The information in this book is arranged as chapters that correspond to the main features of the Logicmaster 90-30/20/Micro software. This manual contains the following chapters:

Chapter 1. Introduction: provides an overview of the features available with the Logicmaster 90-30/20/Micro programming software. After you read chapter 1, refer to the chapters that describe the functions you want to use.

Chapter 2. Operation: explains what you need to know to install and start up the software. It also explains the format of the software screens.

Chapter 3. Program Editing: describes program entry and edit features.

Chapter 4. Reference Tables: describes how to display tables of reference values, change formats in reference tables, force references, and override bit-oriented references.

Chapter 5. PLC Control and Status: describes how to control and modify the operation of a connected PLC. These features include how to change the PLC operating state, display and access PLC privilege levels, display and clear PLC and I/O faults, display PLC and program memory usage, display configured reference sizes, and modify sweep parameters.

Chapter 6. Programmer Setup: explains how to set up the programmer for communication with the PLC, and how to select the programmer operating mode.

Chapter 7. Program Folders: describes how to create, select, rename, modify, copy, or delete program folders.

Chapter 8. Program Utilities: describes how to use program utility functions to transfer programs and tables between the programmer and the PLC, to compare programs and data in the programmer with programs and data in the PLC, and to clear PLC memory. This chapter also describes how to transfer a program, register table, and configuration between the PLC and EEPROM (or Flash), and to compare a program, register table, or configuration in the PLC with a program, register table, or configuration in EEPROM.

Chapter 9. Print Functions: describes how to use the print functions to enter printer parameters, create files containing information to be printed, and print copies of programs, reference tables, and display screens.

Chapter 10. I/O Configuration: describes how to use the configuration software to configure I/O modules.

Chapter 11. CPU Configuration: describes how to use the configuration software to set the operating characteristics of the CPU.

This manual contains the following appendices:

Appendix A. Programming Lesson: provides a sample programming lesson with simple instructions for creating a program folder, creating a program, entering a variable declaration, adding ladder logic to the program, printing the program, and exiting the programmer.

Appendix B. Configuration Lesson: provides a sample configuration lesson with simple instructions for creating a program folder and configuring various modules.

Appendix C. Programmer Environment Setup: describes how to modify the setup parameters.

Appendix D. User Command Menu: describes how to maintain a file of DOS-executable commands outside of the Logicmaster software packages.

Appendix E. Instruction Mnemonics: lists mnemonics that can be typed to display program instructions while searching through or editing a program.

Appendix F. Key Functions: lists the special keyboard assignments used for the Logicmaster 90 software.

Appendix G. Files Created with Logicmaster 90-30/20/Micro Software: describes the files created by Logicmaster 90-30/20/Micro software that are associated with the program or folder.

Appendix H. Common User Errors: describes common problems which may occur and the corrective action to take.

Appendix I. Variable Declaration Table Import/Export Using Comma Separated Variable (CSV) Format: describes the format and requirements you need to consider when using the importing or exporting feature for Variable Declarations.

Related Publications

Series 90™-30/20/Micro Programmable Controllers Reference Manual (GFK-0467).

Logicmaster™ 90 Series 90-30 and 90-20 Important Product Information (GFK-0468).

Series 90™-30 Programmable Controller Installation Manual (GFK-0356).

Series 90™-20 Programmable Controller Installation Manual (GFK-0551).

Series 90™-30 I/O Module Specifications Manual (GFK-0898).

Series 90™ Programmable Coprocessor Module and Support Software User's Manual (GFK-0255).

Series 90™ PCM Development Software (PCOP) User's Manual (GFK-0487).

MegaBasic™ Programming Language Reference Manual (GFK-0256).

CIMPLICITY™ 90-ADS Alphanumeric Display System User's Manual (GFK-0499).

CIMPLICITY™ 90-ADS Alphanumeric Display System Reference Manual (GFK-0641).

Alphanumeric Display Coprocessor Module Data Sheet (GFK-0521).

Series 90™-30 High Speed Counter User's Manual (GFK-0293).

Workmaster® II PLC Programming Unit Guide to Operation (GFK-0401).

Series 90™-30 and 90-20 PLC Hand-Held Programmer User's Manual (GFK-0402).

Series 90™-30 Programmable Controller Axis Positioning Module (APM)—Standard Mode User's Manual (GFK-0840).

Series 90™-30 Programmable Controller Axis Positioning Module (APM)—Follower Mode User's Manual (GFK-0781).

Series 90™-30 Genius Communications Module User's Manual (GFK-0412).

Series 90™-30 Enhanced Genius™ Communications Module User's Manual (GFK-0695).

Genius Communications Module Data Sheet (GFK-0272).

Series 90™ PLC Serial Communications User's Manual (GFK-0582).

Series 90™ Programmable Controller RS-422/RS-485 to RS-232 Converter Data Sheet (GFK-0550).

Series 90™-30 Genius™ Bus Controller User's Manual (GFK-1034).

Series 90™ Micro Programmable Logic Controller User's Manual (GFK-1065).

We Welcome Your Comments and Suggestions

At GE Fanuc Automation, we strive to produce quality technical documentation. After you have used this manual, please take a few moments to complete and return the Reader's Comment Card located on the next page.

David Bruton
Sr. Technical Writer

CUT ON DOTTED LINE

READER'S COMMENTS

GFZ-0050A

We invite your comments and welcome suggestions to make this manual more useful.

Publication No. ____________ Date of Publication ____________ Today's Date ____________

GENERAL COMMENTS:

	Improve	Acceptable	Good	Excellent
Contents	☐	☐	☐	☐
Organization	☐	☐	☐	☐
Accuracy	☐	☐	☐	☐
Clarity	☐	☐	☐	☐
Completeness	☐	☐	☐	☐
Examples/Illustrations	☐	☐	☐	☐
Referencing/Indexing	☐	☐	☐	☐
Readability	☐	☐	☐	☐

DETAILED COMMENTS: (Correct, expand, etc. – Please be specific.)

Page No.	Comment

Other suggestions for improving this document: ____________

____

As compared to other manufacturers of a similar product, how would you rate this document ?

Superior ☐ Comparable ☐ Inferior ☐ Don't Know ☐

Commments: ____________

____

Are you interested in subscribing to a documentation update plan? Yes ☐ No ☐

APPLICATION INFORMATION:

Indicate the type of user/reader function that you most nearly represent:

☐ System Designer	☐ Programmer
☐ Distributor	☐ Maintenance
☐ OEM	☐ Operator
☐ Installation	☐ Other (Please Specify) ____________

Type of Equipment: ☐ Series 90-70 ☐ Series 90-30 ☐ Series 90-20 ☐ Series Six
☐ Series Five ☐ Series One ☐ Genius I/O ☐ Other ____________

Comments concerning your specific application: ____________

____

____

____

From:

Name:	
Title:	
Company:	
Address:	
City/State/Zip:	
Telephone:	

Fold Here

No Postage
Necessary
If Mailed
In The
United States

BUSINESS REPLY MAIL
FIRST CLASS MAIL PERMIT NO. 995 CHARLOTTESVILLE, VA

POSTAGE WILL BE PAID BY ADDRESSEE:

ATTENTION MANAGER TECHNICAL PUBLICATIONS
GE Fanuc Automation North America Inc
P O BOX 8106
CHARLOTTESVILLE VA 22907-6063

Fold Here

Chapter 1 Introduction

Section 1: Product Overview

Logicmaster 90-30/20/Micro programming software is part of a family of products used to configure and program the full line of Series 90-30, Series 90-20, and Micro programmable controllers.

Note

In this manual, the WSI version of Logicmaster 90-30/20/Micro software refers to using a Work Station Interface Board in the programmer to provide serial communication between the programmer and the attached PLC. The standard serial communications version of Logicmaster 90-30/20/Micro software refers to using ports COM1, COM2, COM3, or COM4 instead of the Work Station Interface Board.

Configuration is the process of assigning logical addresses, as well as other characteristics, to the hardware modules in the system. It may be done either before or after programming, using the configuration software; however, it is recommended that configuration be done first.

Programming consists of creating an application program for a PLC. Because Series 90-30 PLCs, Series 90-20 PLCs, and Micro PLCs have a common instruction set, all can be programmed using this software.

What You Will Need

To run Logicmaster 90-30/20/Micro software, you will need:

- A computer with a hard disk:
  - ☐ A Workmaster® II industrial computer with a 101-key keyboard, or
  - ☐ A personal computer with an Intel 80386 or higher processor and a minimum of 2 Megabytes of memory, or
  - ☐ A Zenith™ Mastersport™ SL notebook computer.
- At least 4 Megabytes of **free** disk space.
- Both the Logicmaster 90-30/20/Micro Release 5 WSI and the Standard Serial COM Port versions require a ***minimum*** of 520 KB (532,480 bytes) of available DOS application memory in order to run. The Standard Serial COM Port version requires either a ***minimum*** of 564 KB (577,536 bytes) of available DOS application memory, or 520 KB (532,480 bytes) of available DOS application memory *and* 42 KB of available High Memory Area, Upper Memory Block or Expanded Memory for the COM port driver (see page 6-6 and following for details about the Standard Serial COM Port version and memory management).

  Both versions require a minimum 1024 KB of Lotus/Intel/Microsoft Expanded Memory (LIM EMS 3.2 or higher) for optimum performance. If additional DOS application memory (also called "low memory") is needed, system software error ID: 0000 EX: 0000 will occur. Remove any unneeded TSR (Terminate and Stay Resident) programs and any unnecessary device drivers from the AUTOEXEC.BAT and CONFIG.SYS files to make more memory available. For additional information, see chapter 2, section 3, of this manual.

™ *Zenith and Mastersport are trademarks of Zenith Data Systems Corporation.*
® *MS-DOS is a registered trademark of Microsoft Corporation.*

The MS-DOS Version Needed

To run Logicmaster 90-30/20/Micro software, MS-DOS® Version 5.0 (or higher) must be installed on your computer.

Logicmaster 90-30/20/Micro software provides foreign keyboard support, depending on the configuration of MS-DOS residing on the host computer. Consult your MS-DOS User's Manual for information on configuring for your country.

Help Screens

Logicmaster 90-30/20/Micro software includes detailed Help screens. These Help screens are loaded onto the hard disk of your programmer during the software installation procedure and are readily accessible. To access the Help screens, press **ALT-H** for help, **ALT-I** for instruction mnemonic help, or **ALT-K** for key help.

Key Functions

Appendix E, "Key Functions," lists the keyboard functions that are active in the Logicmaster 90-30/20/Micro software environment. Appendix E also contains a perforated Help card which can be removed from this manual. This information may also be displayed on the programmer screen by pressing **ALT-K** to access key help.

Before You Begin

You will find Logicmaster 90-30/20/Micro programming software easy to use and to understand. When you are ready to begin, turn to chapter 2. It will tell you:

- How to install the software in your computer.
- How to start up the software.
- How to use your computer's keyboard to perform special programming functions.
- How to enter data, move the cursor, and read the status information on your screen.

After starting up the software, try the short programming lesson in appendix A and the configuration lesson in appendix B.

When you are ready to use the programming software to create a program, monitor a system, or perform any of its other functions, you will find instructions in other chapters of this book.

Section 2: Configuration Software

CPU Configuration

Configuration software is used to display and modify the characteristics of the CPU (such as memory allocation, PLC ID, and time of day). Section 3 of chapter 10 explains how to complete the CPU configuration for your system. This will set up or change the system features described below.

PLC Time and Date

The PLC maintains the current time and date. These settings can be displayed and changed using the CPU configuration function in Model 331 or higher CPUs. A time-of-day clock is not available in CPU Models 321, 323, 311, 313, 211, nor in Micro CPUs Models 004 and lower.

PLC Memory Allocation

You can display the current memory allocated to both discrete references and register references.

Fault Display and Clearing

When the programmer is monitoring a PLC, it displays faults that are stored in the PLC fault table. All faults are identified by time, date, and location. Faults can be cleared from the fault table display.

Note

CPU Models 321, 323, 311, 313, 211, and Micro CPUs Models 004 and lower do not support a time-of-day clock. Entries for CPU date and time are displayed as 00-00 00:00:00 in the fault tables.

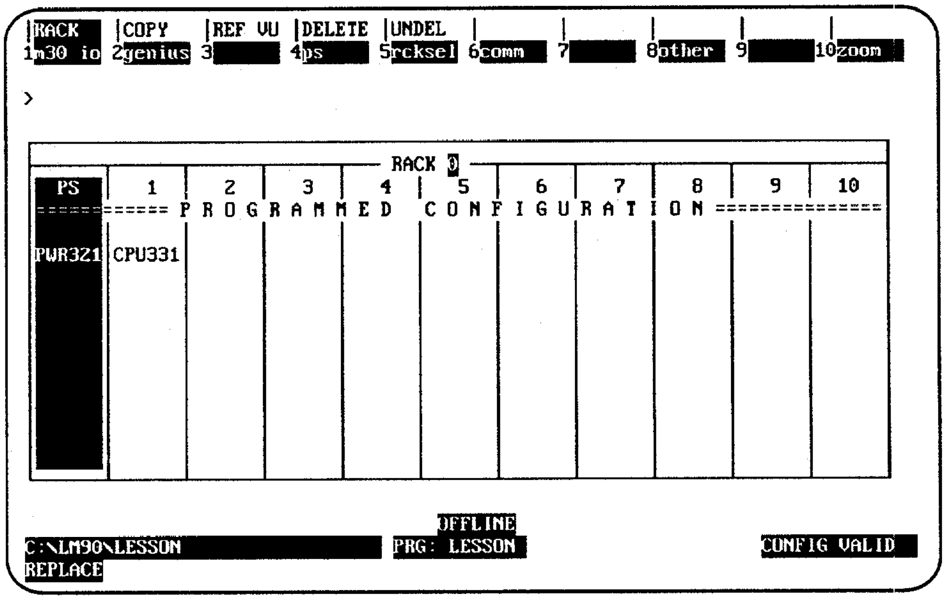
System Response to Faults

To assure safe operation of the control system, the PLC must be able to respond appropriately to certain types of faults. Fatal faults cause the CPU to set fault references and then go to **STOP** mode. Diagnostic faults cause the CPU to set fault references, but the PLC keeps operating.

I/O Configuration

The I/O configuration function is used to describe the modules that are present in the PLC racks, to assign logical addresses, and select options for individual modules. These logical addresses are independent of physical location or function. Chapter 10 explains how to complete the I/O configuration for your system.

The I/O configuration rack screen represents the appearance of the Series 90-30 I/O rack. Use the Next and Previous page keys, or the Up and Down cursor keys, to display another rack. Then use the Left and Right cursor keys to move the cursor to the slot to be displayed or configured.



```
|RACK    |COPY    |REF VU  |DELETE  |UNDEL   |        |        |        |        |
1m30 io  2genius  3        4ps      5rcksel  6comm    7        8other   9        10zoom

>

                                   RACK 0
  PS   |  1  |  2  |  3  |  4  |  5  |  6  |  7  |  8  |  9  |  10
============ P R O G R A M M E D   C O N F I G U R A T I O N ==============
PWR321 |CPU331

                                     OFFLINE
C:\LM90\LESSON                       PRG: LESSON                CONFIG VALID
REPLACE
```

To complete the I/O configuration, you will:

1. Select the module present in each slot.
2. Assign each module a reference address. The configuration software automatically supplies the next highest reference address for each module; however, this address may be changed.
3. For some modules, you may also select options, such as the counter type for a High Speed Counter Module.

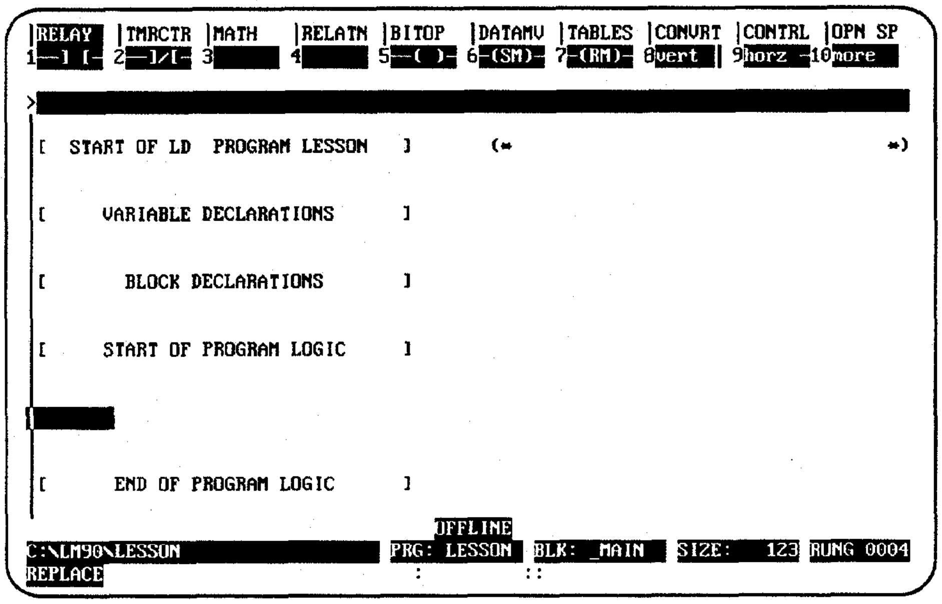
Editing features make it easy to copy, move, replace, delete, or undelete configurations.

Section 3: Programming Software

Creating or Editing a Program

Chapter, "Program Editing," describes how to create and edit programs.

The basic elements of a program are shown when the programming screen is first selected:



```
|RELAY  |TMRCTR |MATH    |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1-] [- 2-]/[- 3        4       5-( )- 6-(SM)- 7-(RM)- 8vert  | 9horz -10more
>
[  START OF LD  PROGRAM LESSON    ]       (*                                   *)

[     VARIABLE DECLARATIONS       ]

[       BLOCK DECLARATIONS        ]

[     START OF PROGRAM LOGIC      ]



[      END OF PROGRAM LOGIC       ]
                                    OFFLINE
C:\LM90\LESSON                     PRG: LESSON BLK: _MAIN  SIZE:    123 RUNG 0004
REPLACE                                 :          ::
```

Marker	Description
Variable Declarations	To access the variable declaration table, move the cursor to this marker and press Zoom (F10). Nicknames and reference descriptions can then be entered in the table.
Block Declarations	A program can include more than one block of logic. Additional blocks, known as subroutines, can be called from other blocks. When that is done, blocks must be declared before they are called. The main block has a block declaration table. This table lists all blocks which are part of the complete program. Blocks do not have block declaration tables. However, blocks can be called from the main block or from any block in the program.
Start/End of Program Logic	All logic is placed between these two markers. To enter logic, place the cursor on the **[END OF PROGRAM LOGIC]** marker and press Insert (F1).

Editing functions include rung insert/edit, select, cut, delete, paste, include, and write. In addition, search and goto functions are provided to position the cursor on a particular rung or element. An optional feature of the search function is the replacement of the search target with a user-specified element and/or reference address.

Chapter 3, section 4, "Program Annotation," describes how annotation can be added to a program to make the program easier to read and understand. Three types of annotation (nicknames, reference descriptions, and rung comments) are supported by Logicmaster 90-30/20/Micro software.

Displaying Tables of Reference Values

Chapter 4, "Reference Tables," explains how to use the reference tables feature to display the current values of program references. If the programmer is connected to a PLC and in **ONLINE** or **MONITOR** mode, the values shown in the table are from the PLC. In **OFFLINE** mode, they are from the current folder. There are separate tables for each type of program reference; for example, all discrete inputs (%I), all discrete outputs (%Q), and all registers (%R). In addition, there are 99 user-defined tables called mixed reference tables.

The format of individual items or an entire reference table can easily be changed to units that are suitable to your application. You can also return a standard reference table to its default format and fill the table locations with zero.

Start/Stop PLC Execution

PLC program execution is started or stopped from the Run/Stop PLC screen, or by pressing ALT-R from any screen. For more information, refer to chapter 5, "PLC Control and Status."

Fault Display and Clearing

When the programmer computer is monitoring an operating PLC system, any faults that have occurred are displayed in one of two fault tables. PLC faults are listed in the PLC fault table. Faults from the I/O system are listed in the I/O fault table. All faults are identified by time, date, and location.

Additional information about each fault can be displayed by positioning the cursor on the fault in the fault table and pressing the Zoom (F10) softkey.

Faults can be cleared from the fault table displays. For information about the PLC and I/O fault tables, refer to chapter 5, "PLC Control and Status."

PLC Sweep Time Display and Change

Chapter 5, "PLC Control and Status," explains how to use the programming software to:

- Display the current CPU sweep time. This is the amount of time required for one complete cycle of program execution, I/O scan, communications windows, and other functions.
- See whether or not **CONSTANT SWEEP TIME** mode is enabled, and read its setting.
- Display the time period of the watchdog timer. This timer is used to shut down the CPU if the sweep time is too long.
- Display the times currently allocated for the programmer window and the system communications window.

PLC and Program Memory Information

Chapter 5, "PLC Control and Status," also explains how to display information about:

- PLC memory size.
- PLC memory used and available.
- Program memory used for logic, data, declarations, and annotation.
- Memory used for configured program references.

Selecting the Programmer Operating Mode

Both the programmer and the configuration software operate in three modes: **OFFLINE**, **MONITOR**, and **ONLINE**. In **OFFLINE** mode, no data transfer takes place between the computer and the PLC. Programs and configuration data may conveniently be developed in **OFFLINE** mode, with or without the computer connected to a PLC. In **MONITOR** mode, if communications have been established between the computer and the PLC, the computer can read data from the PLC but may not transfer data to it. With communications established in **ONLINE** mode, programs and other data can be transferred between the PLC and the computer.

If you are using a Workmaster or CIMSTAR I industrial computer, you may configure the Logicmaster 90-30/20/Micro software to use the keyswitch to select the operating mode. For those computers without a keyswitch, or if the keyswitch is not enabled, mode selection can be made by:

- Pressing the ALT and M keys simultaneously. Repeatedly pressing **ALT-M** switches the operating mode from **OFFLINE** to **MONITOR** to **ONLINE** and then back to **OFFLINE**.
- Going to the Programmer Setup screen (Shift-F7) and selecting an operating mode. For more information on using the Programmer Setup screen to select the operating mode, refer to chapter 6, "Programmer Setup."

System Security

Security for configuration and program functions consists of:

- A range of four privilege levels for the PLC, which may be protected using passwords.
- A software lock that can be applied to individual program folders.
- A selectable operating mode.
- A software lock that can be applied to individual subroutine blocks.

Passwords are a configurable feature of the Series 90-30, 90-20, and Micro PLC. Their purpose is to provide different levels of access privilege for the PLC when the programmer is in **ONLINE** mode. Passwords may not be set, or the password access (privilege) level may not be changed, when the programmer is in **OFFLINE** mode. The use of passwords is optional and may be set up using the status functions in the configurator. Passwords may be used to restrict changing I/O and PLC configuration data, changing programs, and clearing faults.

Chapter 3, section 8, "Subroutine Blocks," describes how individual subroutine blocks may be locked and unlocked.

Chapter 5, "PLC Control and Status," describes the programmer's privilege levels and explains how to enter passwords. Chapter 5 also describes how passwords are specified.

Program folders can be protected to prevent accidental changes to program and configuration information. Each program folder can be either "locked" or "unlocked". After a folder is locked, its contents cannot be changed or deleted. For more information on locking program folders, see chapter 7, "Program Folders."

Setting up PLC Communications using a WSI Board

For the WSI version of Logicmaster 90-30/20/Micro software, the serial port on the Work Station Interface (WSI) Board in the programmer provides serial communication between the programmer and the attached PLC. The COMSET serial port setup function is used to configure the WSI serial port, and to save or recall those configurations from disk files. Refer to chapter 6, "Programmer Setup," for instructions on WSI serial port setup.

Setting up PLC Serial Communications for Standard Serial COM Ports

For the standard serial communications version of Logicmaster 90-30/20/Micro software, the COM1, COM2, COM3, or COM4 serial port may be set up to provide serial communications between the programmer and the attached PLC. The COMSET serial port setup is used to configure the COM1, COM2, COM3, or COM4 serial port, and to save or recall those configurations from disk files. Refer to chapter 6, "Programmer Setup," for instructions on PLC communications serial port setup.

Program Folders

Each program and the corresponding configuration is assigned to a subdirectory called a program folder. Both the configuration software and the programming software use a set of program utility functions to create and maintain program folders. Chapter 7, "Program Folders," explains how to use the program folder functions to:

- Select another program folder.
- Create a new program folder.
- Delete an unneeded program folder.
- Rename a program folder.
- Make a backup copy of the current program folder.
- Restore a program folder with its backup copy.
- Clear the contents of the current program folder.
- Lock/unlock a program folder.
- Copy the contents of one program folder into the current program folder.

Application programs and related files can be stored on the same hard disk as the software. They can also be copied *using Logicmaster folder utilities* to floppy disks for portability or for independent storage.

Transferring Programs

The program utility functions, described in chapter 8, "Program Utilities," are used to:

- Transfer programs and configuration between the PLC and the programmer.
- Clear part or all of the program and/or configuration from PLC memory.
- Compare a program or configuration in the PLC with the current folder to determine if they are the same.
- Transfer a program, register (%R) table, and configuration between the PLC and EEPROM.
- Compare a program, register (%R) table, or configuration in the PLC with EEPROM.

Printing Programs and Configuration

The programming software includes a complete set of print functions. You can select the contents and format of the printout. In addition, you can print cross references for the references used in the program.

You can also print copies of your system configuration by using the print function in the configuration software. Refer to chapter 9, section 5, "Print Configuration," for information about printing configuration screens.

Chapter 2 Operation

This chapter explains what you will need to know to install and start up the Logicmaster 90-30/20/Micro software. It also explains keyboard use and content of the Logicmaster 90-30/20/Micro software screens.

Chapter 2 contains the following sections:

Section 1: Hardware Setup (WSI Version)

Installing the Work Station Interface Board

The Work Station Interface (WSI) Board provides a high-performance serial interface between a Series 90-30 or Series 90-20 PLC and the programmer for the WSI version of Logicmaster 90-30/20/Micro software.

There are two Work Station Interface Boards available; one is for an AT-type computer, and the other is for a PS/2-type computer. The WSI Board is included as part of a package with Logicmaster 90-30/20/Micro programming software. When a Workmaster II computer is ordered as the programming device, the WSI Board is installed at the factory. For installation instructions on other programming devices, consult the computer manufacturer's instructions for option boards.

Grounding

Be sure the computer has a ground connection in common with the CPU rack. This is usually done by connecting the programmer computer to the same power source (with the same ground reference point) as the rack.

Warning

If the programmer is not connected as described above, damage to the Work Station Interface Board can occur. Erratic control operation may also result. If the programmer is online to an operating system, possible erratic operation may cause conditions which are hazardous to personnel and equipment.

Cabling

The communications cable that connects the computer to the PLC is a shielded twisted pair cable with a 3-pin D connector on each end. The maximum length for this cable is 50 feet. After installing the Work Station Interface Board in your computer, connect the computer to the PLC by first attaching the cable to the Work Station Interface Board and then to the port on the rack power supply.

Section 2: Hardware Setup (Standard Serial Communications Version)

If you have the Logicmaster 90-30/20/Micro version of software which communicates with the PLC using the standard serial communication ports, a serial card with COM1, COM2, COM3, or COM4 must be installed in the computer in order to communicate with the PLC. Refer to GFK-0356, *Series 90-30 Programmable Controller User's Manual,* or GFK-0551, *Series 90-20 Programmable Controller User's Manual,* for instructions on establishing a serial connection between the Series 90-30, Series 90-20, or Micro PLC serial port and the serial port on the programming computer, without having a Work Station Interface Board installed in the computer.

Section 3: Software Installation

To use Logicmaster 90-30/20/Micro software, it must be installed on the programmer computer's hard disk. MS-DOS Version 5.0 or higher must already exist on the hard disk.

The Logicmaster 90-30/20/Micro installation procedure creates three subdirectories on the hard disk:

Software Version	Subdirectories Created
WSI	\LM90 \LM90\P30 \LM90\C30
Standard serial communications	\LM90 \LM90\P30S \LM90\C30S

AUTOEXEC.BAT and CONFIG.SYS Files

Before starting to install Logicmaster 90-30/20/Micro software, check the content of your hard disk root directory to see whether the files CONFIG.SYS and AUTOEXEC.BAT are present. Operation of Logicmaster 90-30/20/Micro software requires these files to be present. If they are not, they are installed automatically. If your hard disk already has these two files, you may be asked during installation whether the install process should modify them. If you prefer, you can edit your existing files for use with Logicmaster 90-30/20/Micro software.

The CONFIG.SYS (System Configuration) file is a short, readable file that describes the configuration of MS-DOS. Different software packages may use different system configurations. For all Logicmaster 90-30/20/Micro applications, the file must contain at least these two lines:

```
  Files = 20
Buffers = 48
```

If you want to check the content of an existing CONFIG.SYS file, you can use the TYPE command.

- The WSI version of Logicmaster 90-70 software requires a minimum of 520 KB (532,480) bytes of available MS-DOS application memory in order to run.

  The Standard Serial COM Port version requires either a ***minimum*** of 564 KB (577,536 bytes) of available DOS application memory, or 520 KB (532,480 bytes) of available DOS application memory *and* 42 KB of available High Memory Area, Upper Memory Block or Expanded Memory for the COM port driver (see page 6-6 and following for details about the Standard Serial COM Port version and memory management).

  Both versions require a minimum 1024 KB of Lotus/Intel/Microsoft Expanded Memory (LIM EMS 3.2 or higher) for optimum performance. If additional DOS application memory (also called "low memory" or "conventional memory") is needed, system software error ID: 0000 EX: 0000 will occur. Remove any unneeded TSR (Terminate and Stay Resident) programs and any unnecessary device drivers from the AUTOEXEC.BAT and CONFIG.SYS files to make more memory available.

The following configurations will assist you in setting up your computer system to run Logicmaster 90-30/20/Micro software optimally on 386/486 computers with a minimum of 2 MB RAM.

LM90-30/20/Micro Software (Standard MS-DOS Version 5.0)	**Configuration**

CONFIG.SYS	DEVICE=C:\DOS\HIMEM.SYS
	DOS=HIGH,UMB
	DEVICE=C:\DOS\EMM386.EXE 1024 RAM X=CE00–CFFF
	FILES=20
	BUFFERS=48

Note

The above EMM386.EXE switch "X=CE00–CFFF" is used to reserve this area of memory for the Series 90-30/20/Micro WSI card. If the computer does not have a WSI card, the "X=CE00–CFFF" is not required; however, the "EMM386.EXE 1024 RAM" specification enhances performance and may be necessary if you have over 537 MB of free hard disk space (see next paragraph below). If you have less than 537 MB of free hard disk space, the special precautions discussed below are unnecessary.

If you have over 537 MB of free hard disk space and you have the statement DEVICE=C:\DOS\EMM386.EXE NOEMS in your CONFIG.SYS, you will receive the following error, **`(00032) An unknown error occurred during install initialization`**. To prevent this error from occurring, you must specify Expanded Memory in your CONFIG.SYS, e.g., DEVICE=C:\DOS\EMM386.EXE 1024 RAM. If you are using a commercial memory manager, such as QEMM™ or 386MAX™, this precaution may not be necessary. (See IPI GFK-0350S which comes with this Logicmaster release for examples of tested configurations using QEMM™ and 386MAX™.)

In addition, some folders may require additional memory. If additional memory is required, system software error ID: 0000 EX: 0000 will be displayed. Check the AUTOEXEC.BAT and CONFIG.SYS files to remove any device drivers and Terminate and Stay Resident (TSR) programs in order to free more RAM. Logicmaster 90-30/20/Micro software does not require the ANSI.SYS device driver.

The AUTOEXEC.BAT file must have **`(drive ID):\LM90`** added to the existing path. **`(drive ID)`** is the letter which corresponds to the hard disk drive where Logicmaster 90-30/20/Micro software is installed.

If you need to run Version 4.01 or later of Logicmaster 90-30/20/Micro software from a disk that does not contain the software, add this to your AUTOEXEC.BAT file:

```
SET $PLCROOT=(drive ID):\LM90
```

™ 386MAX is a trademark of Qualitas, Inc.
™ QEMM-386 and QRAM are trademarks of Quarterdeck Office Systems.

Running Other Software with the CONFIG.SYS File for Logicmaster 90-30/20/Micro Software

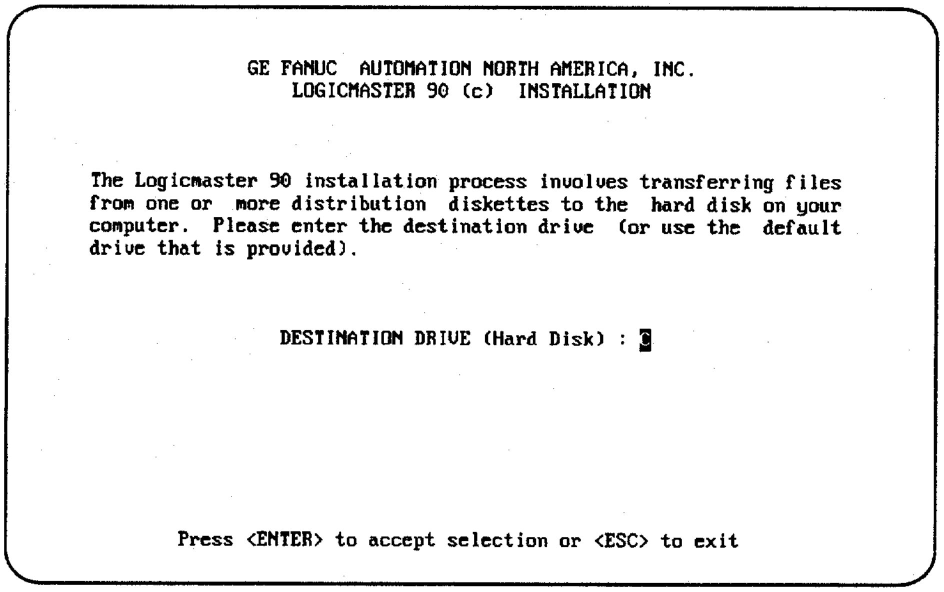
Other types of software may require different entries in the CONFIG.SYS file. It is not always possible to combine the requirements for multiple software packages in one CONFIG.SYS file. In that case, you must maintain multiple versions of the CONFIG.SYS file with different file names, or if you are using MS-DOS version 6 or higher, you can create multiple boot options within one config.sys—see your DOS manual for information on this feature. Your MS-DOS manual contains other information about the CONFIG.SYS file that may be useful to you.

If you have loaded device drivers for special devices or a local area network, which terminate and stay resident, these programs may need to be removed before installing and running the Logicmaster 90-30/20/Micro software. The MS-DOS CHKDSK command may be used to determine available memory (bytes free).

Logicmaster 90-30/20/Micro software does not require the ANSI.SYS device driver; it may be removed in order to obtain more memory.

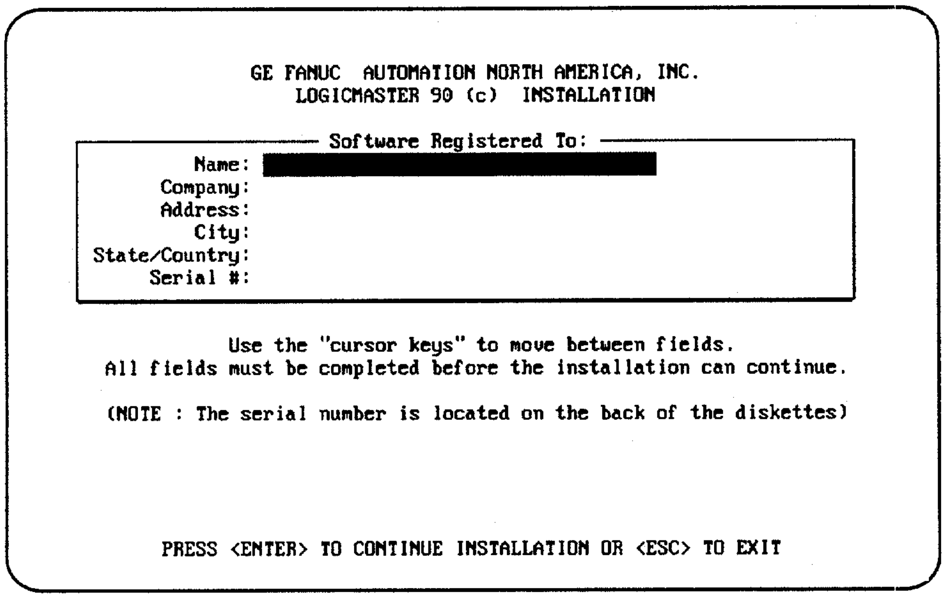
Installation Instructions

1. Boot up the computer using MS-DOS.
2. Be sure CONFIG.SYS has files set to at least 20, i.e., FILES=20.
3. Remove the write-protect tab from the disk.
4. Insert the Logicmaster 90-30/20/Micro software disk into the computer's disk drive.
5. At the MS-DOS prompt, enter the designation of the disk drive followed by a colon. For example, if the disk is in drive A, type **`A:`** and press the **`Enter`** key.
6. Begin the installation procedure by typing **`INSTALL`** and pressing the **`Enter`** key. The following screen is displayed.



```
              GE FANUC  AUTOMATION NORTH AMERICA, INC.
                  LOGICMASTER 90 (c)  INSTALLATION


   The Logicmaster 90 installation process involves transferring files
   from one or  more distribution  diskettes to the  hard disk on your
   computer.  Please enter the destination drive  (or use the  default
   drive that is provided).


                 DESTINATION DRIVE (Hard Disk) : 


          Press <ENTER> to accept selection or <ESC> to exit
```

7. Specify the hard disk drive and press the **Enter** key, or just press the **Enter** key if the default is correct. The following screen is displayed if this is the first installation of this disk.



```
                 GE FANUC  AUTOMATION NORTH AMERICA, INC.
                     LOGICMASTER 90 (c)  INSTALLATION

         ─────────────── Software Registered To: ──────────────
                   Name:
                Company:
                Address:
                   City:
          State/Country:
               Serial #:

                Use the "cursor keys" to move between fields.
        All fields must be completed before the installation can continue.

        (NOTE : The serial number is located on the back of the diskettes)



            PRESS <ENTER> TO CONTINUE INSTALLATION OR <ESC> TO EXIT
```

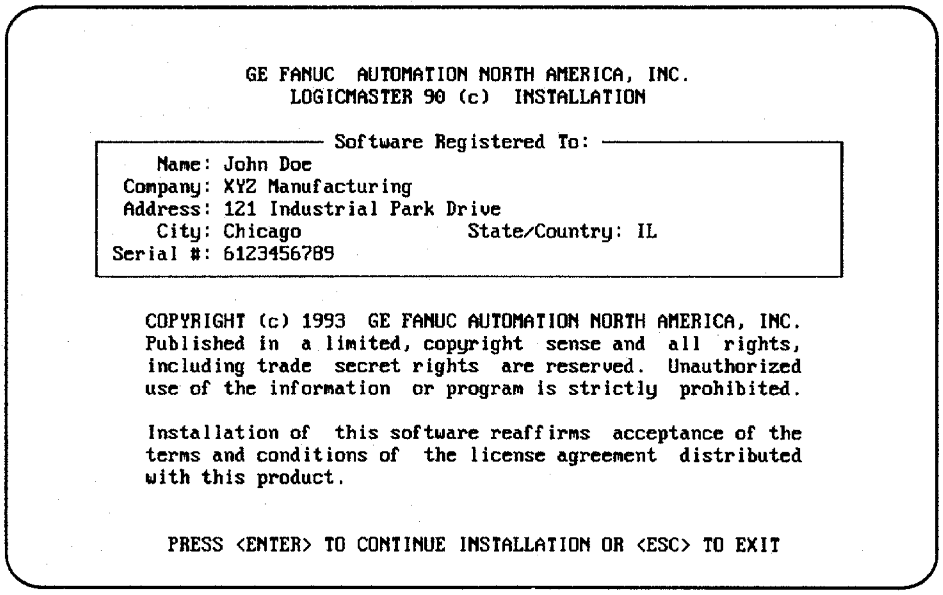
Note

The ***Serial Number*** field must contain the serial number from your registration card or the back of the distribution disks.

8. Use the cursor keys to move between fields. Each field is validated as you move off the field. You must fill in all of the fields and press the **Enter** key in order for the installation process to continue. If any fields are empty or invalid when the **Enter** key is pressed, the first invalid field is highlighted and an error message is displayed. The system will prompt you to confirm that the registration information is correct as displayed.

   If the registration data is not correct, press: **N** (No) and correct the registration information. When the registration data is correct, press: **Y** (Yes). The data is then encoded and written onto the master distribution disk. This screen is displayed until the registration data is successfully written to the master disk, or until the installation program is aborted by pressing the **Escape** key. You may print this screen by pressing the **Print Screen** key.

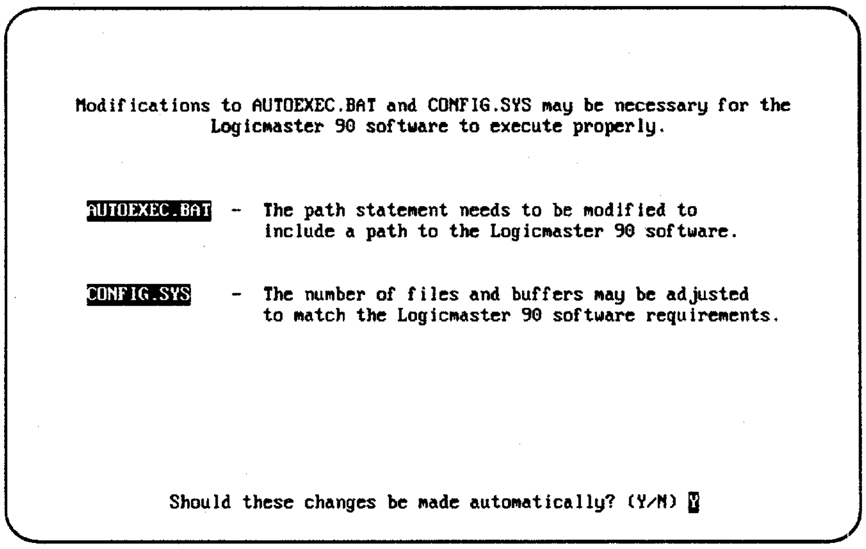
9. When the registration data is successfully written to the master disk, the installation process can begin. The following screen is displayed after the registration data has been entered, or on future Logicmaster installations of the same software package. (Registration is not required once the data is written to the master disk.) Read the licensing agreement.



```
                GE FANUC  AUTOMATION NORTH AMERICA, INC.
                   LOGICMASTER 90 (c)  INSTALLATION

     ---------------------- Software Registered To: ----------------------
        Name: John Doe
     Company: XYZ Manufacturing
     Address: 121 Industrial Park Drive
        City: Chicago                 State/Country: IL
    Serial #: 6123456789

         COPYRIGHT (c) 1993  GE FANUC AUTOMATION NORTH AMERICA, INC.
         Published in  a limited, copyright  sense and  all  rights,
         including trade  secret rights  are reserved.  Unauthorized
         use of the information  or program is strictly  prohibited.

         Installation of  this software reaffirms  acceptance of the
         terms and conditions of  the license agreement  distributed
         with this product.

           PRESS <ENTER> TO CONTINUE INSTALLATION OR <ESC> TO EXIT
```

This information is displayed each time an installation is performed. Pressing the **Enter** key after reading this screen means you agree to comply with the stated terms. Press the **Escape** (**ESC**) key to terminate INSTALL and return to MS-DOS.

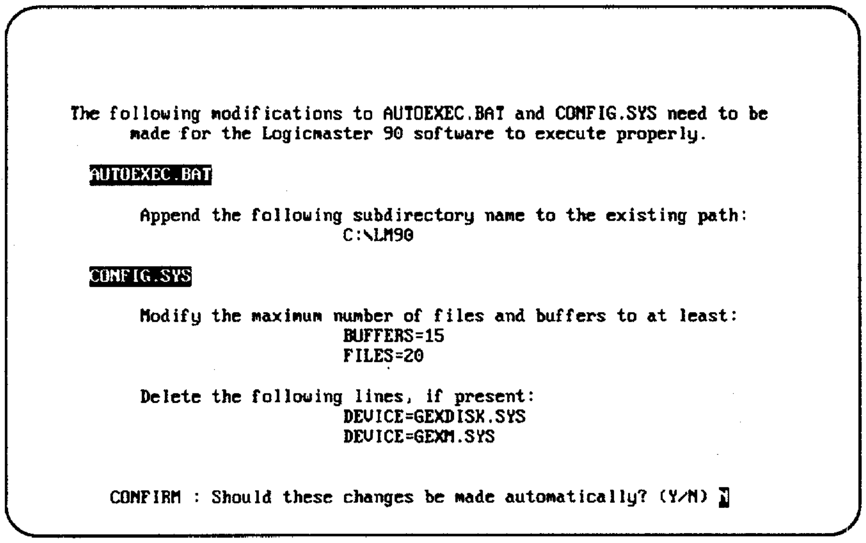
10. First, the Installation procedure creates the \LM90 directory in the root directory. INSTALL checks the files AUTOEXEC.BAT and CONFIG.SYS in the root directory of the hard disk. These files must contain certain commands to ensure that Logicmaster 90-30/20/Micro software executes properly. If neither file exists, they are both created. If either file already exists, INSTALL will ask if the files should be automatically modified.



```
Modifications to AUTOEXEC.BAT and CONFIG.SYS may be necessary for the
             Logicmaster 90 software to execute properly.


AUTOEXEC.BAT  -  The path statement needs to be modified to
                 include a path to the Logicmaster 90 software.

CONFIG.SYS    -  The number of files and buffers may be adjusted
                 to match the Logicmaster 90 software requirements.




              Should these changes be made automatically? (Y/N) Y
```

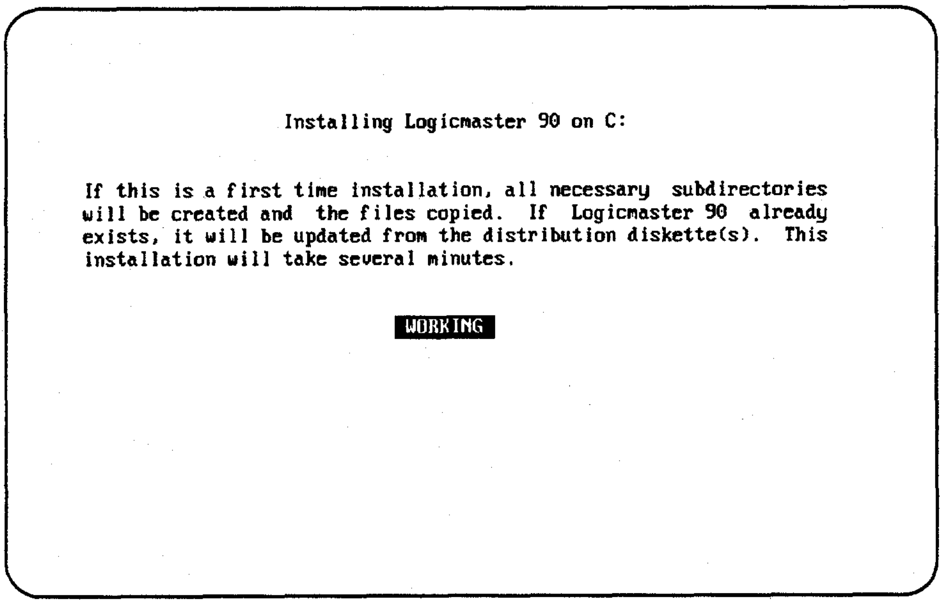
If you want the AUTOEXEC.BAT and CONFIG.SYS files to be automatically modified, enter **Y** (Yes) or press the **Enter** key. If there were already versions of those files in the root directory, they will be renamed to AUTOEXEC.L90 and CONFIG.L90, respectively.

If you already have AUTOEXEC.BAT and/or CONFIG.SYS files and plan to edit them yourself, enter **N** (No). The following screen is displayed:



```
The following modifications to AUTOEXEC.BAT and CONFIG.SYS need to be
      made for the Logicmaster 90 software to execute properly.
 AUTOEXEC.BAT

      Append the following subdirectory name to the existing path:
                           C:\LM90
 CONFIG.SYS

      Modify the maximum number of files and buffers to at least:
                           BUFFERS=15
                           FILES=20

      Delete the following lines, if present:
                           DEVICE=GEXDISK.SYS
                           DEVICE=GEXM.SYS

    CONFIRM : Should these changes be made automatically? (Y/N) N
```

11. If you change your mind and want INSTALL to automatically update the AUTOEXEC.BAT and CONFIG.SYS files, enter **`Y`** (Yes). Otherwise, enter **`N`** (No) or press the **`Enter`** key.



```
                    Installing Logicmaster 90 on C:

   If this is a first time installation, all necessary  subdirectories
   will be created and  the files copied.  If  Logicmaster 90  already
   exists, it will be updated from the distribution diskette(s).  This
   installation will take several minutes.

                               WORKING
```

12. INSTALL attempts to create three subdirectories under the root directory on the hard disk, and transfers the appropriate files to those subdirectories. If the subdirectories already exist, any files in them are deleted and the new files installed. If folders exist, they are not deleted.
13. INSTALL will then prompt you to insert any other disks. After all the files have been transferred, the final installation screen is displayed.
14. The computer must now be re-booted in order to complete the installation process. After removing the last disk used during the installation process, press **`CTRL-ALT-Delete`** to re-boot the computer. When the MS-DOS prompt is displayed, enter **`LM90`** to start up the software.

Programmer Setup

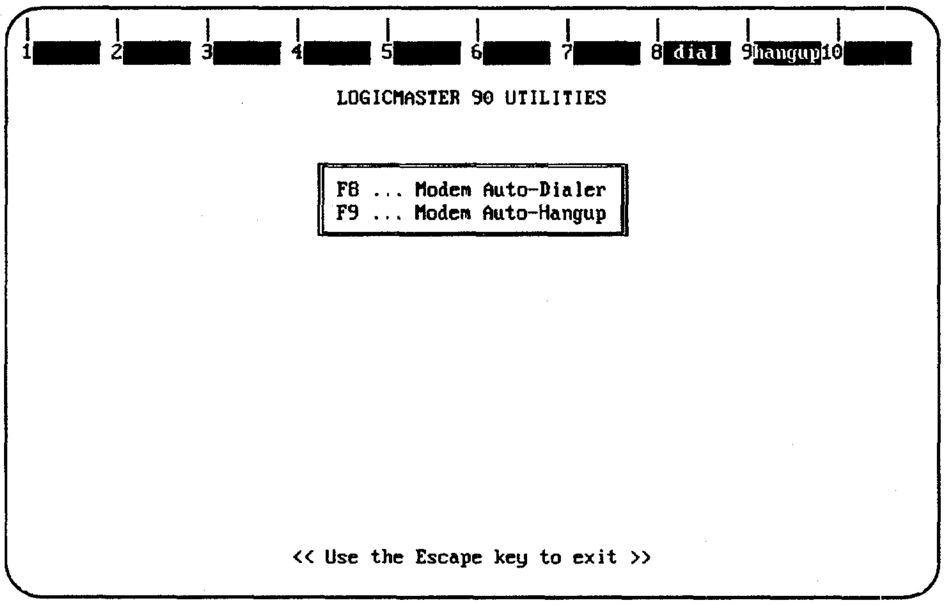
A default setup file is created during installation. Only in special circumstances will this file need to be changed. Refer to appendix C, "Programmer Environment Setup," for instructions on modifying the setup parameters.

Using a Modem

Logicmaster 90-30/20/Micro software has a modem auto dial feature which can be accessed from the main menu of Series 90 PLCs and functions by selecting Logicmaster 90 **Utilities (F7)**.

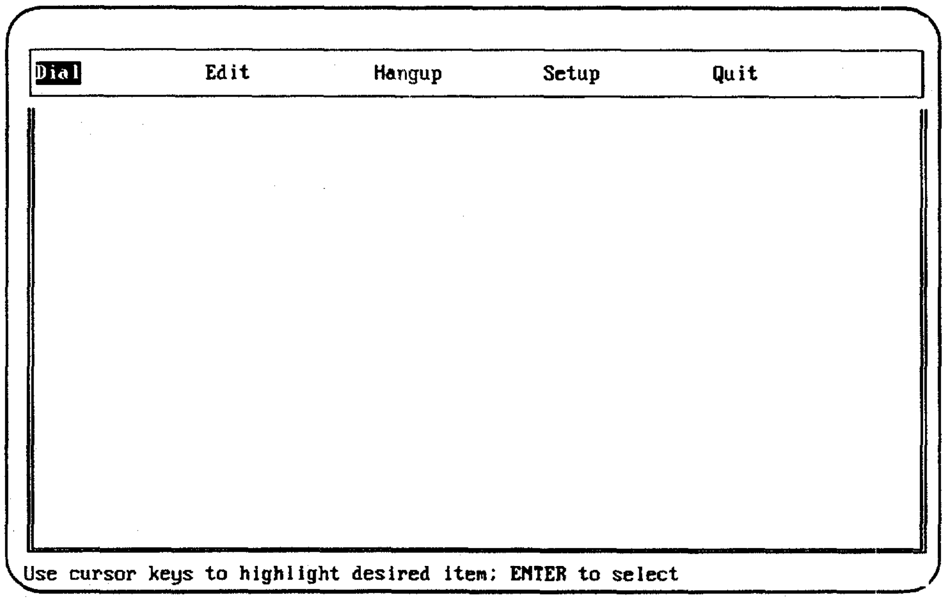
Note

The Logicmaster 90-30/20/Micro modem auto dial feature only supports COM1 and COM2.



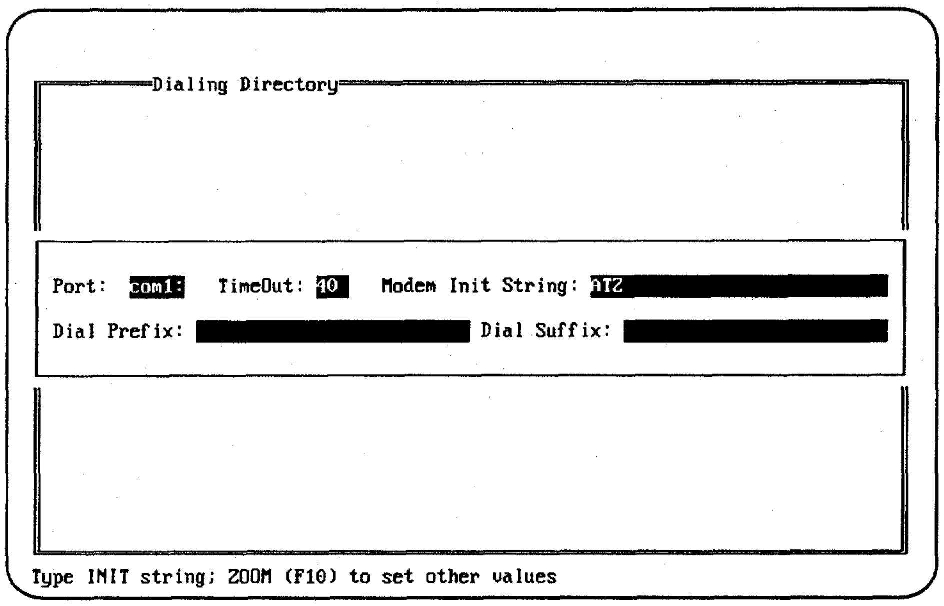
```
LOGICMASTER 90 UTILITIES

F8 ... Modem Auto-Dialer
F9 ... Modem Auto-Hangup

<< Use the Escape key to exit >>
```

1. Press **F8** to select the modem auto dial feature.



```
Use cursor keys to highlight desired item; ENTER to select
```

2. To set the modem parameters, move the cursor to ***Setup*** and press the **Enter** key.

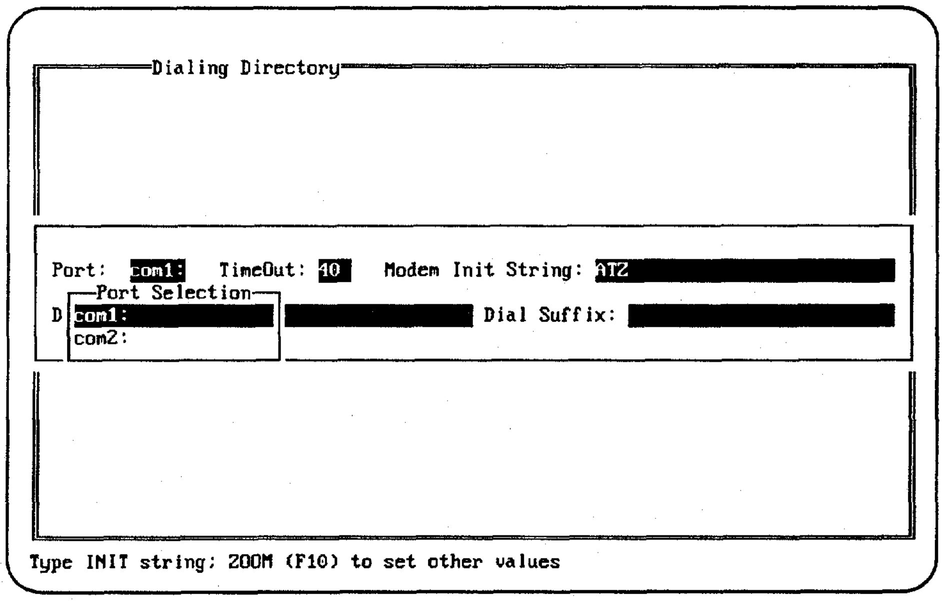


```
Dialing Directory

Port: com1:   TimeOut: 40   Modem Init String: ATZ
Dial Prefix:                    Dial Suffix:

Type INIT string; ZOOM (F10) to set other values
```

A. Complete the fields on the screen displayed, using the Enter key to move among the fields and the right/left cursor keys to move within each field. Press **Zoom** (**F10**) to display other values, as shown in the following screen.

Note

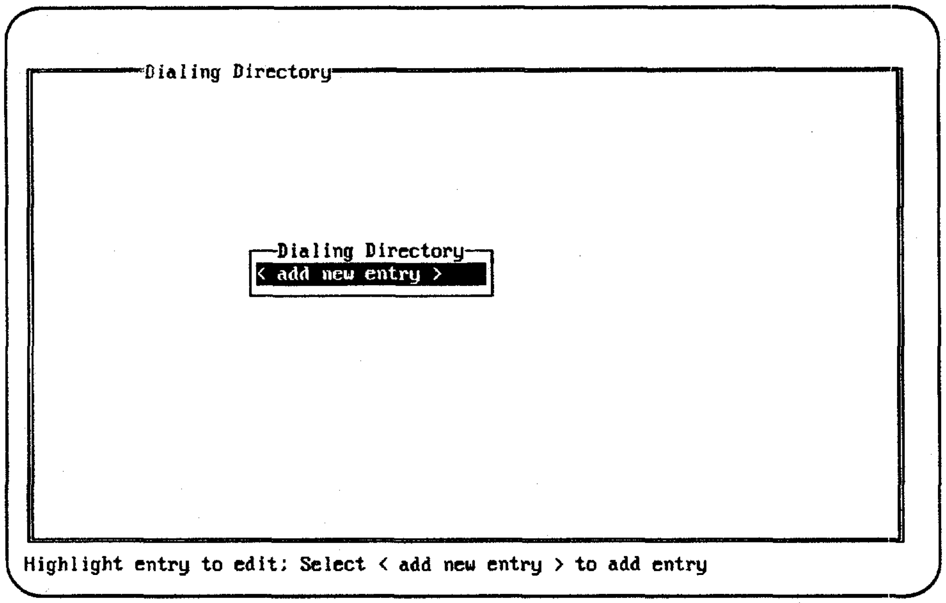
You can add any non-numerical characters that your modem accepts for the dial prefix or suffix, e.g., commas to add pauses (particularly useful when trying to connect to a number in a different country). See your modem user's manual for acceptable non-numerical characters.



```
Dialing Directory

Port: com1:   TimeOut: 40   Modem Init String: ATZ
  ┌Port Selection┐
D │com1:         │                Dial Suffix:
  │com2:         │

Type INIT string; ZOOM (F10) to set other values
```

B. Press the **Enter** key to select a parameter for each field. When all the fields are complete, press the **Escape** key.

3. To edit an entry or enter a new listing in the directory, move the cursor to ***Edit*** and press the **Enter** key.



```
Dialing Directory

        Dialing Directory
        < add new entry >

Highlight entry to edit; Select < add new entry > to add entry
```

   A. Highlight the entry you wish to edit, or select: ***< add new entry >*** and press the **Enter** key to add a new listing.

   B. Complete the fields on the screen displayed, using the **Enter** key to move among the fields and the right/left cursor keys to move within each field. Press **Zoom** (**F10**) to display the choices for each parameter, cursor to one of the choices, and then press **F10** again or the **Enter** key to select the parameter. When all the fields are complete, press the **Escape** key.

4. To dial a number, press the **Enter** key with the cursor on ***Dial***. Use the cursor keys to select an entry, and then press the **Enter** key to dial the number.

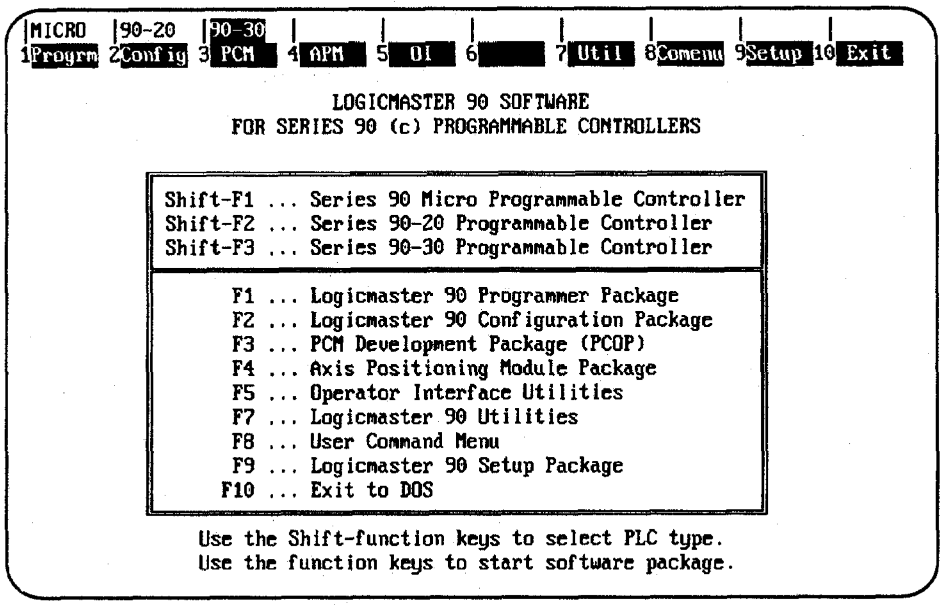
5. Move the cursor to ***Hangup*** and press the **Enter** key to abort the call and hang up the modem.

6. To exit the modem, cursor to ***Quit*** and press the **Enter** key, or press **ALT-F8**.

Section 4: Startup/Exit

To start up Logicmaster 90-30/20/Micro software:

1. At the MS-DOS prompt, type **LM90** and press the **Enter** key. The menu of Series 90 PLCs and functions is displayed.



|MICRO |90-20 |90-30 | | | | | | |
1Progrm 2Config 3 PCM 4 APM 5 OI 6 7 Util 8Comenu 9Setup 10 Exit

```
                    LOGICMASTER 90 SOFTWARE
          FOR SERIES 90 (c) PROGRAMMABLE CONTROLLERS

   Shift-F1 ... Series 90 Micro Programmable Controller
   Shift-F2 ... Series 90-20 Programmable Controller
   Shift-F3 ... Series 90-30 Programmable Controller

         F1 ... Logicmaster 90 Programmer Package
         F2 ... Logicmaster 90 Configuration Package
         F3 ... PCM Development Package (PCOP)
         F4 ... Axis Positioning Module Package
         F5 ... Operator Interface Utilities
         F7 ... Logicmaster 90 Utilities
         F8 ... User Command Menu
         F9 ... Logicmaster 90 Setup Package
        F10 ... Exit to DOS

       Use the Shift-function keys to select PLC type.
       Use the function keys to start software package.
```

Note

Beginning with Release 3, a teach file can be used to enter the Logicmaster 90-30/20/Micro software. This entry teach file will contain the keystrokes previously taught while entering the programming or configurator software. Information on using a teach file to enter the Logicmaster 90-30/20/Micro software can be found in chapter 2, section 5, "Keyboard Functions."

2. Use the function keys to select the programming, configuration, or PCM configuration software.

3. If only one version of Logicmaster 90-30/20/Micro software (either WSI or standard serial communications) is installed, it will be run automatically. If both versions are installed, the Logicmaster 90 Setup Package (F9) can be used to select the version to run. Refer to appendix C, "Programmer Environment Setup," for instructions on modifying the programmer environment setup.

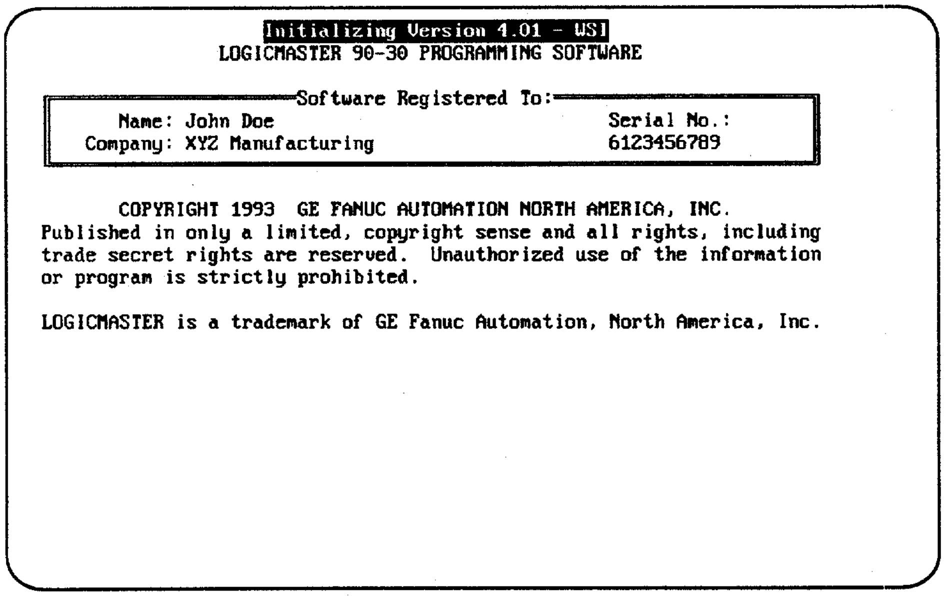
Starting the Programming or Configuration Software

To load the programming software, select Program (**F1**) from the menu of Series 90 PLCs and functions, shown above. If you wish to load the configuration software, select Config (**F2**), or select PCM (**F3**) to load the PCM configuration software. (For information on the PCM development software package, refer to the *Series 90 PCM Development Software (PCOP) User's Manual*, GFK-0487.)

The Series 90-30, Series 90-20, and Micro PLCs use the same Logicmaster 90-30/20/Micro software package for programming. The default PLC type can be selected using the appropriate shift-function key sequence, as described below, but any of the three PLC types can be configured using the configuration package.

- If the configuration file does not already exist in the selected program folder and the Micro (**Shift-F1**) key is pressed before booting the Logicmaster software, the reference defaults and CPU will be that of a Micro CPU.
- If the configuration file does not already exist in the selected program folder and the 90-20 (**Shift-F2**) key is pressed before booting the Logicmaster software, the reference defaults and CPU will be that of a Model 211 CPU.
- If the configuration file does not already exist in the selected program folder and the 90-30 (**Shift-F3**) key is pressed before booting the Logicmaster software, the reference defaults and CPU will be that of a Model 331 CPU.

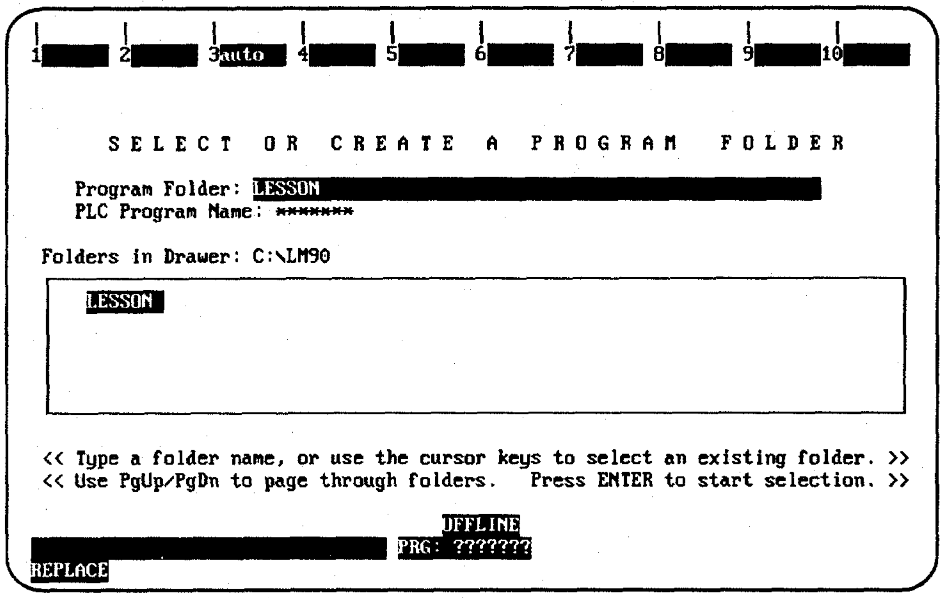
After selecting **Program** (**F1**) or **Config** (**F2**), the following screen appears:



```
                    Initializing Version 4.01 - WS1
                 LOGICMASTER 90-30 PROGRAMMING SOFTWARE

 ═══════════════════════Software Registered To:═══════════════════════
     Name: John Doe                                 Serial No.:
  Company: XYZ Manufacturing                        6123456789

       COPYRIGHT 1993  GE FANUC AUTOMATION NORTH AMERICA, INC.
Published in only a limited, copyright sense and all rights, including
trade secret rights are reserved.  Unauthorized use of the information
or program is strictly prohibited.

LOGICMASTER is a trademark of GE Fanuc Automation, North America, Inc.
```

When the Logicmaster 90-30/20/Micro software powers up, it attempts to automatically select a program folder:

- If there is a program folder beneath the drawer last selected, whose name matches the program name in the attached PLC, that folder is selected.
- If the current directory is a program folder and its name matches the program name in the attached PLC, that folder is selected.
- If there is a program folder beneath the current directory, whose name matches the program name in the attached PLC, that folder is selected.
- If the current directory is a program folder, it is selected.
- Otherwise, the initial select screen is displayed with the name of the last folder selected in the *Folder* field.



```
SELECT OR CREATE A PROGRAM FOLDER

  Program Folder: LESSON
  PLC Program Name: ********

Folders in Drawer: C:\LM90

  LESSON

<< Type a folder name, or use the cursor keys to select an existing folder. >>
<< Use PgUp/PgDn to page through folders.   Press ENTER to start selection. >>

                                   OFFLINE
                                   PRG: ???????
REPLACE
```

Note

At serial baud rates of 1200 or less, the folder cannot be automatically selected during startup. Once in the programming software, however, you may use the auto function to automatically select a folder. (Refer to the information on "Automatic Folder Selection" in chapter 7, "Program Folders.")

Enter a name of up to seven characters for the program folder. (The name is also used for the program.) After entering a name, press the **Enter** key. If you need more information about program folders, refer to chapter 7, "Program Folders."

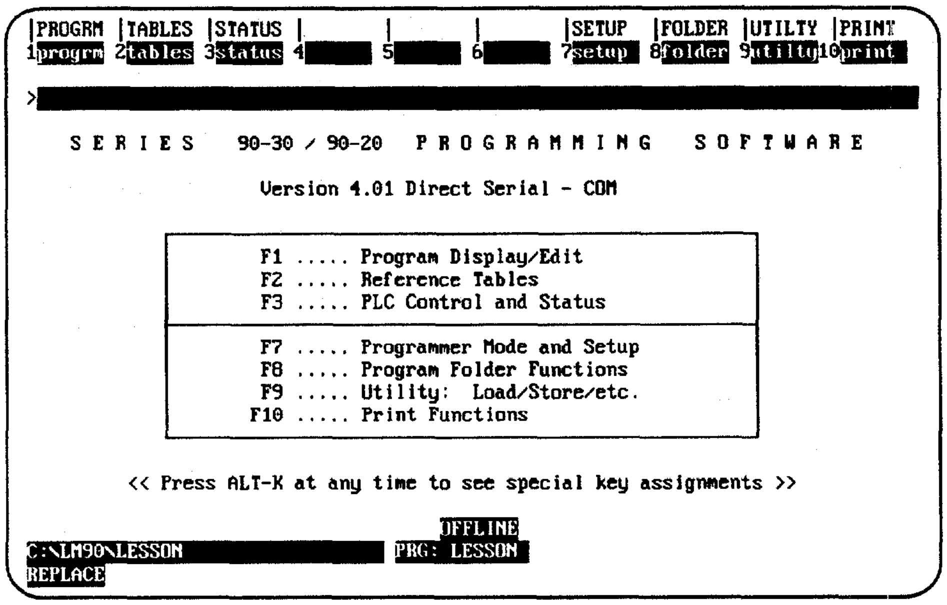
Exiting (Quitting) the Programming or Configuration Software

To exit the software at any time, press **CTRL-Break** and confirm the prompt, or repeatedly press the **Escape** key at each prompt. You should always exit the software before turning the computer off; otherwise, some changes may be lost.

If you respond to the exit prompt ("Exit Logicmaster 90 Package? (Y/N)") by entering a **1** instead of a **Y** (Yes) or **N** (No), a bookmark feature saves the current context. The next time you run the software, you will return to the same general location when you re-enter the software.

Programming Software Main Menu

If you selected **Program (F1)** from the menu of Series 90 PLCs and functions, the Programming Software main menu shown below is displayed. This menu is used to access the primary functions of Logicmaster 90-30/20/Micro programming software.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1progrm 2tables 3status 4       5       6       7setup  8folder 9utilty10print
>

   S E R I E S    90-30 / 90-20    P R O G R A M M I N G    S O F T W A R E

                    Version 4.01 Direct Serial - COM

               F1 ..... Program Display/Edit
               F2 ..... Reference Tables
               F3 ..... PLC Control and Status

               F7 ..... Programmer Mode and Setup
               F8 ..... Program Folder Functions
               F9 ..... Utility:  Load/Store/etc.
              F10 ..... Print Functions

          << Press ALT-K at any time to see special key assignments >>

                                             OFFLINE
C:\LM90\LESSON                              PRG: LESSON
REPLACE
```

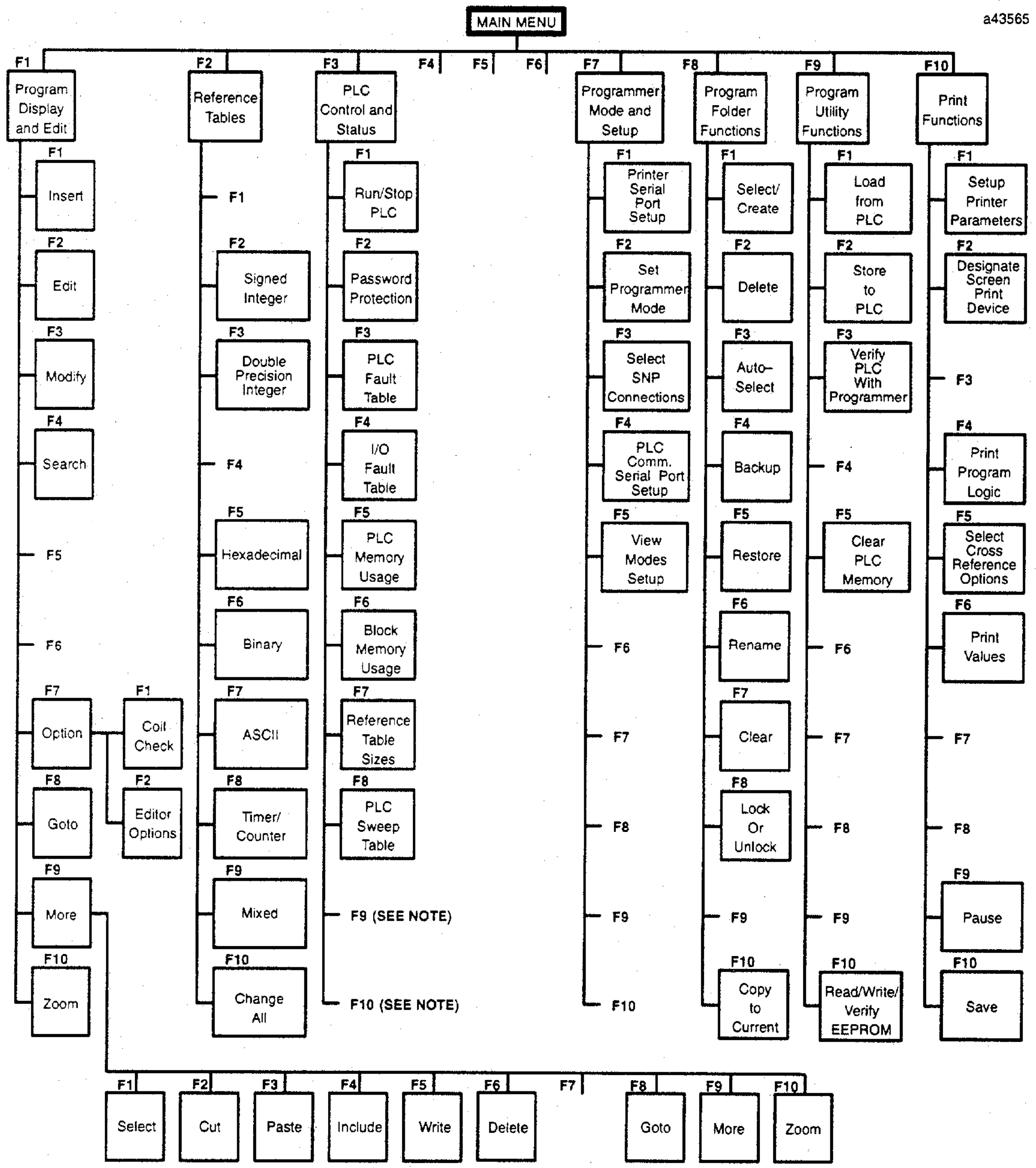
Note

For the WSI version of Logicmaster 90-30/20/Micro software, **WSI** is displayed after the version number in the screen shown above. For the standard serial communications version of software, **COM** is displayed after the version number.

Function Key	Function	Description
F1	Program	Create or edit a program or monitor program logic. Chapter 3 describes the program display and explains how to create or edit a program.
F2	Tables	Display and change reference data. To select a particular reference table, enter the reference type on the command line before pressing **F2**. Chapter 4 describes monitoring data and explains how to change tables and variable data.
F3	Status	Select the status functions. These functions include displays of I/O faults and PLC faults. The memory configuration and the current PLC access level can also be viewed. Chapter 5 describes these functions.
F7	Setup	Display and change serial port setup and other programmer configuration. Chapter 6 describes these functions.
F8	Folder	Create, select, clear, rename, delete, lock, or back up a program folder. Chapter 7 explains how to use these functions.
F9	Utility	Load, store, or verify a program, or clear PLC memory. Chapter 8 explains how to use the utility functions.
F10	Print	Print a program folder. Chapter 9 describes the print functions.

The function keys remain active after selecting a programming function. You can go directly from one programming function to another without returning to the main menu by simultaneously pressing the **Shift** key and the desired function softkey.

The bottom of the main menu screen contains three lines of status information. Chapter 2, section 6, "Screen Format," explains the content of the screen's status area.

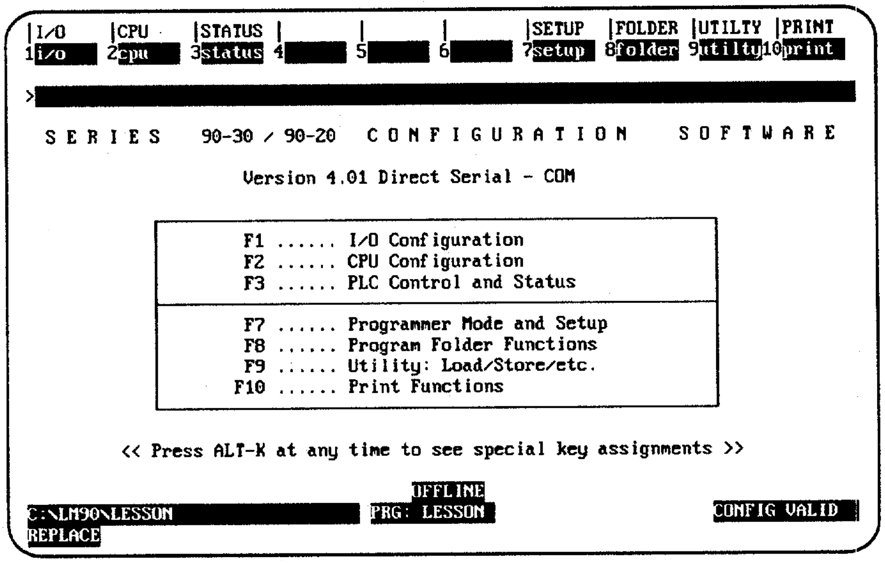


NOTE: **Press Insert Rung (F1), Edit Rung (F2), or Modify Rung (F3) to display the programming instruction menus.**
When Password Protection (F2) is pressed (PLC Control and Status functions), F9 will become the OEM softkey.
When PLC Fault Table (F3) or I/O Fault Table (F4) is pressed (PLC Control and Status functions), F9 will become the Clear softkey and F10 will become the ZOOM softkey.

Figure 2-1. Programming Software Menu Tree

Configuration Software Main Menu

If you selected **Configuration (F2)** from the menu of Series 90 PLCs and functions, the Configuration Software main menu shown below is displayed. The menu lists the primary functions of the software. The first three menu entries are used to select PLC configuration functions. The remaining entries are used to select support functions.



```
|I/O      |CPU      |STATUS |       |       |       |SETUP   |FOLDER  |UTILTY  |PRINT
1i/o      2cpu      3status 4       5       6       7setup   8folder  9utilty  10print
>

S E R I E S    90-30 / 90-20    C O N F I G U R A T I O N    S O F T W A R E

                    Version 4.01 Direct Serial - COM

             F1 ...... I/O Configuration
             F2 ...... CPU Configuration
             F3 ...... PLC Control and Status

             F7 ...... Programmer Mode and Setup
             F8 ...... Program Folder Functions
             F9 ...... Utility: Load/Store/etc.
             F10 ..... Print Functions

     << Press ALT-K at any time to see special key assignments >>

                                     OFFLINE
C:\LM90\LESSON                       PRG: LESSON                 CONFIG VALID
REPLACE
```

Note

For the WSI version of Logicmaster 90-30/20/Micro software, **WSI** is displayed after the version number in the screen shown above. For the standard serial communications version of software, **COM** is displayed after the version number.

Function Key	Function	Description
F1	I/O	I/O configuration is the process of describing to the software the content of a PLC system's rack(s). Chapter 10 tells how to complete I/O configuration.
F2	CPU	CPU configuration sets the operating characteristics of the CPU. Chapter 11 describes the CPU configuration.
F3	Status	The status function displays I/O faults and PLC faults. You will also use this function to display PLC information such as memory usage and to assign passwords. Chapter 5 describes these displays.
F7	Setup	The programmer setup function sets up the computer's serial ports, current operating mode, and PLC connection. Refer to chapter 6 for information on the setup function.
F8	Folder	Program folder functions are a group of file-handling utilities which you will use to create, delete, back up, and limit access to configuration and program files. Chapter 7 describes these functions.
F9	Utility	Program utilities are used to transfer information between the computer and the PLC. An additional program utility is used to clear memory in the PLC. Chapter 8 explains how to use the program utilities.
F10	Print	Print functions are used to generate configuration printouts and to set up a destination for screen prints. Chapter 9 describes these print functions.

The functions are selected using the function keys shown on the top line of the screen. You can go directly from one function to another without returning to the main menu by using the shift function keys.

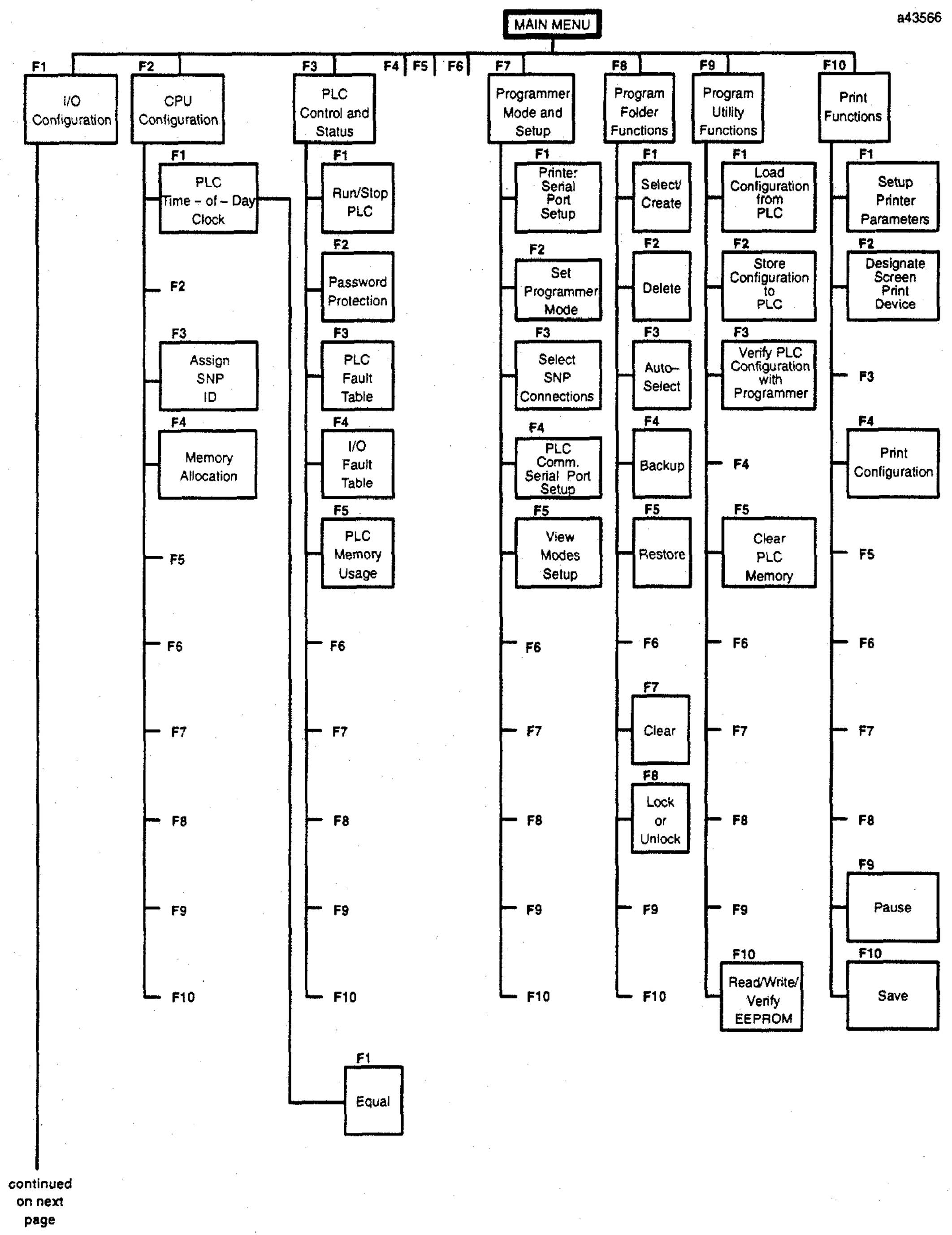


Figure 2-2. Configuration Software Menu Tree

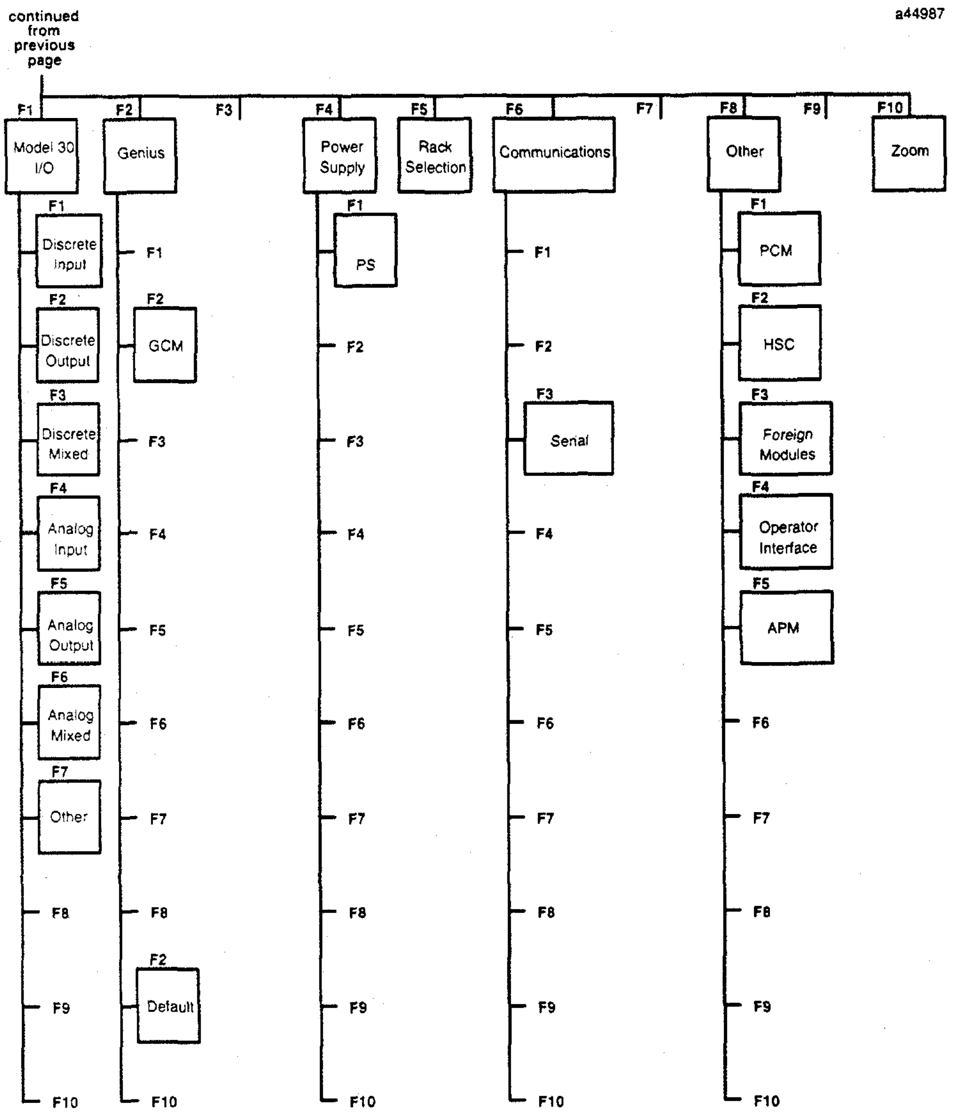


Figure 2-2. Configuration Software Menu Tree (cont'd)

Section 5: Keyboard Functions

This section describes the keyboard functions that are active in the Logicmaster 90-30/20/Micro software environment. It also describes playback functions, which can be used to assign sequences of frequently-used keystrokes to a file for simple recall later on.

Keyboards Supported

Logicmaster 90-30/20/Micro programming software can be used with the 83-key or 101-key keyboard for an IBM PC personal computer, or a Workmaster, Workmaster II, or CIMSTAR I industrial computer. Other types of keyboards for the IBM PC-XT, PC-AT, or IBM-compatible computer may work with Logicmaster 90-30/20/Micro software, but have not been tested. The 91-key keyboard for the Workmaster computer, developed for use with Logicmaster 6 programming software, would be difficult to use since Logicmaster 90-30/20/Micro software was designed to use with standard keyboards.

Key Functions

Appendix E, "Key Functions," lists the keyboard functions that are active in the Logicmaster 90-30/20/Micro software environment. Appendix E also contains a perforated Help card which can be removed from this manual. This information may also be displayed on the programmer screen by pressing **ALT-K** to access key help.

Keyboard Macros (Teach Mode)

Logicmaster 90-30/20/Micro software can be instructed to record sequences of keystrokes and play them back with a single keystroke. These sequences of keystrokes are referred to as keystroke macros or teach sequences.

A combination of keystrokes might represent a series of frequently used functions or part of a program that you want to duplicate. An example of a simple keystroke macro would be the sequence of keystrokes needed to create the following line of logic:

```
|
| ???????                                                            ???????
|--| |-------------------------------------------------------------------( )-
|
```

The auto-next highest reference address function, described at the end of chapter 3, section 9, "Rung Edit," provides another example of using a keystroke macro.

The keystrokes that make up a keystroke macro are stored in a file named KEYx.DEF, where x is a digit in the set 0 to 9 that you choose when you create the keystroke macro. This file is called a teach file; it is located in the current program folder.

Creating a Teach Sequence

Follow the steps below to create a keystroke macro:

1. Select the starting point in Logicmaster 90-30/20/Micro software at which you wish your macro to begin. You must remember this starting point when you execute or play back the macro you are about to record. For example, if you are editing ladder diagram logic, you must ensure that a condition such as whether you were in **EDIT** mode when you started recording holds when you are playing back the macro.
2. It is often simplest to begin the macro from an easy-to-remember starting point such as the main menu, the Print menu, or the Edit Program commands. When you wish to begin recording keystrokes, press and hold the **ALT** key while pressing the **T** key. Then, select a playback file by pressing one of the key combinations between **ALT-0** and **ALT-9**. If you select a key combination that has been used previously in the current folder, the software will ask you if you want to overwrite it. Logicmaster 90-30/20/Micro software will now record all the keystrokes that you make.
3. When you have completed your sequence of keystrokes, press **ALT-Q** to exit **TEACH** mode. The software will stop recording keystrokes and create a teach file.

Playing Back a Teach Sequence

Before replaying the keystroke macro, be sure that the Logicmaster 90-30/20/Micro software is in the state it was in when you started recording the macro. Then, press the **ALT-digit** combination you were prompted for when you started recording. The software will rapidly execute the keystrokes you recorded as though they were entered from the keyboard.

The only keyboard input recognized during playback is **ALT-A**, which causes the Logicmaster 90-30/20/Micro software to ask you if you want to stop the playback.

If an error condition is encountered during playback, an error message is displayed, playback is paused, and you will be prompted to continue or stop the key sequence.

Pausing the Playback of a Teach Sequence

A pause may be entered in a keystroke macro. If you are recording a sequence of keystrokes and need to pause playback to examine something on the screen, press **ALT-Z** at the point where you would like playback to pause. This may be done as many times as you like while recording a macro. A message will be displayed telling you that a user prompt key (a pause) has been recorded.

When the Logicmaster 90-30/20/Micro software encounters the ALT-Z key combination during playback, it displays a prompt and waits until the **space bar** is pressed to continue.

Using a Teach File to Run Software from an MS-DOS Batch File

Beginning with Release 3.02, Logicmaster 90-30/20/Micro software may be run using a teach file, which contains keystrokes previously taught while entering the Logicmaster 90-30/20/Micro programming or configuration software packages. This feature allows **LM90** to be invoked from an MS-DOS batch file, to play back keystrokes to perform desired function(s), and to return to the batch file to execute the next command. This can be helpful when executing multiple print requests.

The keystroke macro is stored as the text file <filename>.tch. The file name must identify either the programming or configuration software package. For example, prg9030.tch could be used as the file name for the programming software and cfg9030.tch could be used for the configurator software. The file extension must be **.tch**.

To begin recording keystroke macro, enter the executable path, at the MS-DOS prompt, which corresponds to the version of software installed on your computer:

Software Version	Executable Path
WSI programmer	c:\lm90\p30\prg9030
WSI configurator	c:\lm90\c30\cfg9030
Standard serial communications programmer	c:\lm90\p30s\prg9030
Standard serial communications configurator	c:\lm90\c30s\cfg9030

Enter the name of the teach file for this macro (e.g., prg9030.tch) and the letter **t** after the path. The **t** indicates **TEACH** mode. For example, to begin recording keystrokes to a teach file named prg9030.tch, enter:

```
C:\lm90\p30\prg9030 prg9030.tch t
```

To end the teach sequence, press **ALT-Q**, or **CTRL-Break** to end the teach sequence and exit the software.

To play back the keystroke macro, enter the path and teach file name, followed by the letter **p** for **PLAYBACK** mode. For example, to play back a macro for the programming software, enter:

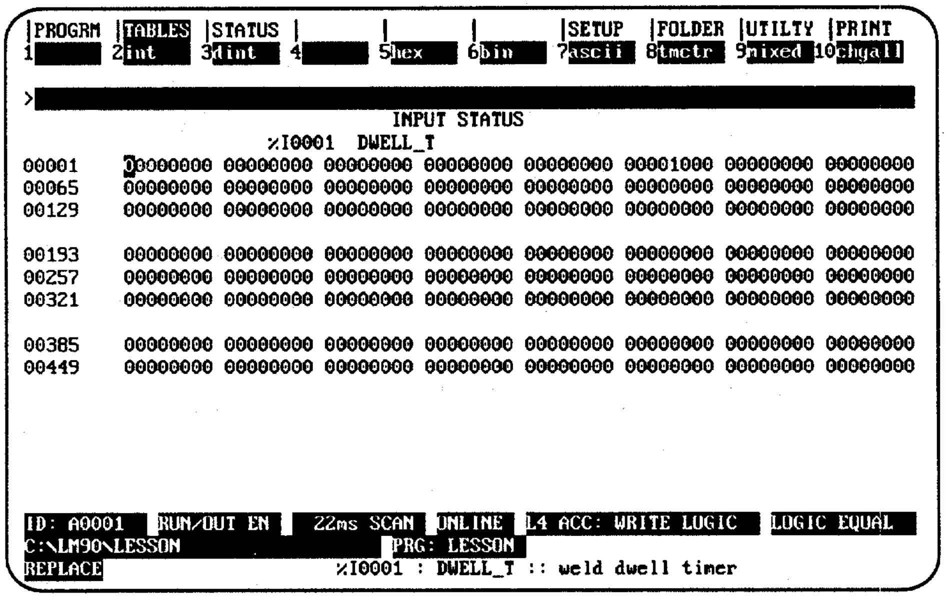
```
C:\lm90\p30\prg9030 prg9030.tch p    or    c:\lm90\p30\prg9030 p
```

To initialize the communications driver for the standard COM port version of Logicmaster 90 software, the batch file must contain the line: **c:\lm90wsil30 i** before the command to create or play back the teach file. The line: **c:\lm90\wsil30 r** must be added after the command to create or play back the teach file in order to remove the communications driver. For example:

```
c:\lm90\wsil30 i
c:\lm90\p30s\prg9030 p
c:\lm90\wsil30 r
```

Section 6: Screen Format

This is a typical bit-oriented reference table display for Logicmaster 90-30/20/Micro software.



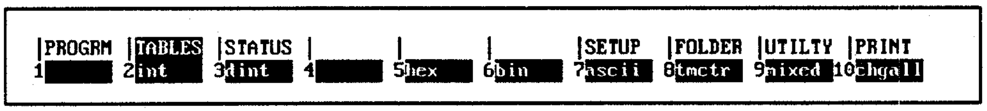
```
|PROGRM |TABLES |STATUS |        |        |        |SETUP   |FOLDER  |UTILTY  |PRINT
1       2int    3dint   4        5hex     6bin     7ascii   8tmctr   9mixed  10chgall

>
                                   INPUT STATUS
                        %I0001  DWELL_T
00001     00000000 00000000 00000000 00000000 00000000 00001000 00000000 00000000
00065     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00129     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

00193     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00257     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00321     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

00385     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00449     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000



ID: A0001   RUN/OUT EN    22ms SCAN  ONLINE  L4 ACC: WRITE LOGIC    LOGIC EQUAL
C:\LM90\LESSON                        PRG: LESSON
REPLACE                       %I0001 : DWELL_T :: weld dwell timer
```

The main part of the screen shows menus, data, and other information related to the function you are currently using.

Function Key Assignments

The top two lines of the screen show the functions that are currently available. The main functions displayed in the upper line are selected by pressing the **`Shift`** key with a function key, **`F1`** through **`F10`**.

The active function appears in a reverse-video block. In the following example, the active main function is reference tables.



```
|PROGRM |TABLES |STATUS |        |        |        |SETUP   |FOLDER  |UTILTY  |PRINT
1       2int    3dint   4        5hex     6bin     7ascii   8tmctr   9mixed  10chgall
```

The lower line displays the secondary functions that can be selected with that main function key active. In the example above, they are the display formats that are available while using reference tables. These formats are selected by pressing a function key, F1 through F10, without using the Shift key.

Message Line

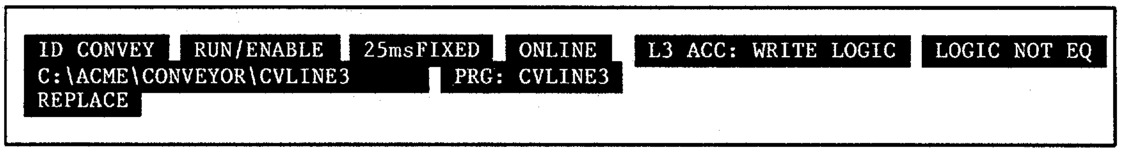
Directly under the function key area is the message line. Errors in command syntax, or those discovered while executing commands or selections, are displayed on the message line. Prompts for additional information required from the user are also displayed on the message line.

Command Line

The command line, identified by the > prompt, displays typed entries, such as data to be used for the table being displayed. It is used to enter instructions, references, or comments.

Status Information

The bottom three lines display information about the status of the programmer, the PLC, the program, and the keyboard. The information displayed will change depending on the programmer mode. For example:

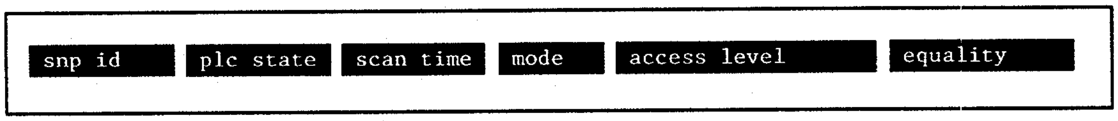


```
ID CONVEY  RUN/ENABLE  25msFIXED  ONLINE   L3 ACC: WRITE LOGIC  LOGIC NOT EQ
C:\ACME\CONVEYOR\CVLINE3        PRG: CVLINE3
REPLACE
```

Status Area	Description
Top line	Displays information about the attached PLC and the programmer.
Second line	Identifies the current program.
Third line	Shows the status of the keyboard; whether Caps Lock, Scroll Lock or Num Lock is active, and whether the keyboard is in **REPLACE** or **INSERT** mode. For some programming functions, the third status line displays additional information, described in later sections of this manual. Items appear in the status information area only when needed.

PLC/Programmer Status: Definitions

The top line of the status area displays information about a PLC and about the programmer. Some of the information displayed reflects selections made using the PLC monitoring functions. For information about using these functions, refer to chapter 5, "PLC Control and Status."



```
snp id    plc state    scan time    mode    access level    equality
```

Items that may appear on this line are explained in the following table. (Items will not appear on the status lines if they are not used for the currently active function.)

Field	Description
SNP ID	The Series Ninety Protocol (SNP) identifier assigned to the PLC.
PLC State	The current status of the CPU: **RUN/OUT EN:** PLC running the logic program, outputs enabled. **STOP/IOSCAN:** PLC stopped, not executing the logic program, scanning I/O. **STOP/NO IO:** PLC stopped, not executing the logic program, no I/O scan. **STOP/FAULT:** PLC stopped due to a fault; check fault tables.
Scan Time	The CPU sweep time in milliseconds. This is followed by the type of the scan, which may be: **SCAN:** Each scan executed as fast as possible. **FIXED:** Constant sweep timer enabled, scan fixed to set time limit. **OVER:** Constant sweep timer enabled, scan exceeds the set time limit.
Mode	The current mode of the programmer connected to the CPU: **OFFLINE:** No communications with PLC, or no PLC attached. **ONLINE:** Actively communicating with the PLC. **MONITOR:** Same as ONLINE, but programmer cannot modify the contents of the PLC.
Access Level	The password access level of the PLC: **LEVEL 4:** Change password, write logic/configuration. **LEVEL 3:** Write logic/configuration, PLC stopped. **LEVEL 2:** Write data, clear fault tables. **LEVEL 1:** Read PLC only.
Equality	This field compares the program in the PLC with the version in the folder: **LOGIC EQUAL:** Both program versions are the same. **LOGIC NOTEQ:** Program versions may be different. **BLOCK EDIT:** The PLC program is the same as that in the current program folder, the current block is being edited online and may be different in the programmer. The block can be stored to the PLC by pressing **ALT-S**.

Selecting the Programmer Operating Mode

Both the programmer and the configuration software operate in three modes: **OFFLINE**, **MONITOR**, and **ONLINE**. In **OFFLINE** mode, no data transfer takes place between the computer and the PLC. Programs and configuration data may conveniently be developed in **OFFLINE** mode, with or without the computer connected to a PLC. In **MONITOR** mode, if communications have been established between the computer and the PLC, the computer can read data from the PLC but may not transfer data to it. With communications established in **ONLINE** mode, programs and other data can be transferred between the PLC and the computer.

If you are using a Workmaster or CIMSTAR I industrial computer, you may configure the Logicmaster 90-30/20/Micro software to use the keyswitch to select the operating mode. For those computers without a keyswitch, or if the keyswitch is not enabled, mode selection can be made by:

- Pressing the **ALT** and **M** keys simultaneously (i.e., press the **Alt** key and hold, then press the **M** key). Repeatedly pressing **ALT** and **M** switches the operating mode from **OFFLINE** to **MONITOR** to **ONLINE** and then back to **OFFLINE**.
- Going to the Programmer Setup screen (**Shift-F7**) and selecting an operating mode. For more information on using the Programmer Setup screen to select the operating mode, refer to chapter 6, "Programmer Setup."

Chapter 3 Program Editing

This chapter describes the program edit features of Logicmaster 90-30/20/Micro programming software. Program display and editing may be protected by passwords. If your system has been set up to use passwords and is in **ONLINE** mode, you may have to enter a password in order to use the functions described in this chapter. Chapter 3 contains the following sections:

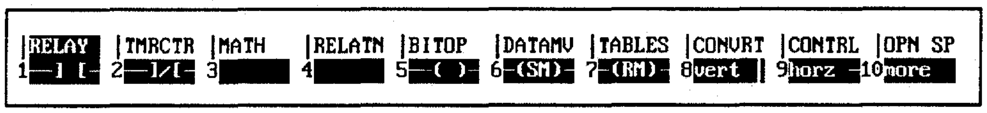
Section	Title	Description	Page
1	Ladder Logic Program Elements	Summarizes the Series 90-30/20/Micro instruction set. Refer to the *Series 90-30/20/Micro Programmable Controllers Reference Manual,* GFK-0467, for additional information on each of these instructions.	3-2
2	Program Format	Describes the format of a ladder logic rung.	3-15
3	Program Entry	Describes how to insert logic elements and functions in a program and three ways to exit from a rung.	3-21
4	Program Annotation	Describes program annotation and how to enter nicknames and reference descriptions.	3-29
5	Variable Declaration Table	Describes how to enter annotation using the variable declaration table. Also describes how to import and export files in SNF format.	3-33
6	Rung Comments	Describes how to insert rung comments.	3-47
7	Changing the Display Mode	Describes how to show reference description information at each symbol or reference address in the program.	3-53
8	Subroutine Blocks	Describes how to use block declarations in a main program block.	3-59
9	Rung Edit	Describes how to modify an existing program by adding rungs, deleting rungs, and editing rungs. This section also describes how to select, cut, paste, include, and write rungs or declarations, and how to use the open space functions to open element spaces in the rung.	3-69
10	Editor Options	Describes options in the program editor, including multiple coil use and automatic insertion of references.	3-95
11	Search Function	Describes how to search for a reference or instruction using the search function.	3-101
12	Online Editing/ Monitoring	Describes how to make changes to or debug a program while the program is running.	3-107

Note

When the program editor is entered, the Logicmaster 90-30/20/Micro software checks the amount of space on the disk containing the program folder. If insufficient space exists to support changes to the program, the folder is temporarily locked. Program logic may be viewed, but not changed. If space is freed up on the disk using MS-DOS, the lock state is automatically cleared when you run Logicmaster 90-30/20/Micro software.

Section 1: Ladder Logic Program Elements

The tables in this section summarize the programming instructions available for Series 90-30 PLCs, Series 90-20 PLCs, and Micro PLCs. You can access all of these instructions from either the **Insert (F1)** or **Edit (F2)** function by selecting one of the main functions displayed in the upper line shown below. Press the **Shift** key and the function key for the function you wish to select. For example, to select one of the math functions, first press **Shift-F3** to display the math functions listed in table 3-3 in this section. Then, select the appropriate math function by pressing the function key for that particular function.


|RELAY |TMRCTR |MATH |RELATN |BITOP |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1—] [- 2—]/[- 3 4 5—( )- 6-(SM)- 7-(RM)- 8vert | 9horz -10more

These function keys provide access to the instructions required to edit ladder diagram rungs:

Function Key	Function	Description	Page
F1	Relay Functions	Select contacts, coils, and links in ladder logic rungs.	3-3
F2	Timer/ Counter Functions	Select on-delay and stopwatch-type timers, up counters, and down counters.	3-5
F3	Math Functions	Select addition, subtraction, multiplication, division,modulo division, and square root functions.	3-6
F4	Relational Functions	Select functions which can compare two numbers: equal, not equal, greater than, greater than or equal, less than, and less than or equal. Or, test a value against a range of numbers.	3-7
F5	Bit Operation Functions	Select functions which can perform comparison and move operations on bit strings: Logical AND, OR, and exclusive OR; logical invert; shift left or right; rotate left or right; bit test, bit set, bit clear, and bit position.	3-8
F6	Data Move Functions	Select basic data move functions: move, block move, block clear, shift register, bit sequencer, and communications request.	3-8
F7	Table Functions	Copy from one array to another; and search for all array values which are equal, not equal, greater than, less than, greater than or equal, or less than or equal to a specified value.	3-9
F8	Conversion Functions	Convert a data item from one number type to another, such as signed integer to 4-digit BCD format or BCD-4 to signed integer format.	3-9
F9	Control Functions	Limit program execution, call a subroutine, enter a comment in a program, and alter the way the CPU executes the program.	3-10

Appendix D, "Instruction Mnemonics," lists the complete mnemonics for each programming instruction. Please refer to the *Series 90-30/20/Micro Programmable Controllers Reference Manual*, GFK-0467, for additional information on each instruction.

Table 3-1. Relay Functions

Key	Instruction	Mnemonic	Function	Description
Selecting Relay Functions (Shift-F1) displays these function key assignments:				
F1	-| |-	&NOCON	Normally Open Contact	A normally open contact passes power if the associated reference is ON.
F2	-|/|-	&NCCON	Normally Closed Contact	A normally closed contact passes power if the associated reference is OFF.
F5	-( )-	&NOCOIL	Normally Open Coil	The associated reference is set ON if the coil receives power.
F6	-(SM)-	&SMLAT	Retentive SET Coil	The associated reference is set ON if the coil receives power. The reference remains set until reset by an -(RM)- coil. Its state is retained through power failure and **STOP-TO-RUN** transition.
F7	-(RM)-	&RMLAT	Retentive RESET Coil	The associated discrete reference is reset OFF if the coil receives power. The reference remains reset until set by an -(SM)- coil. Its state is retained through power failure and **STOP-TO-RUN** transition.
F8	vert |	&VE	Vertical Link	A vertical link with no contact function or reference.
F9	horz –	&HO	Horizontal Link	A shunt; also acts as a delete function.
F10	more			Additional relay functions you can select.

Table 3-1. Relay Functions (cont'd)

Key	Instruction	Mnemonic	Function	Description
Pressing More (F10) displays these additional relay function key assignments:				
F1	---<+>	&COILCTD	Continuation Coil	If power to the coil is ON, the continuation coil sets the next continuation contact ON. If power is OFF, the continuation coil sets the next continuation contact OFF.
F2	<+>---	&CONCTD	Continuation Contact	The continuation contact passes power to the right if the preceding continuation coil is set ON.
F3	–(/M)–	&NCMCOIL	Negated Retentive Coil	The associated discrete reference is set ON if the function does not receive power. The state is retained through power failure and **STOP-TO-RUN** transition.
F4	–(/)–	&NCCOIL	Negated Coil	The associated discrete reference is set ON if the coil does not receive power.
F5	–(M)–	&NOMCOIL	Retentive Coil	The associated discrete reference is set ON if the coil receives power. The state is retained through power failure and **STOP-TO-RUN** transition.
F6	–(S)–	&SLAT	SET Coil	The associated discrete reference is set ON if the coil receives power. It remains set until reset by an –(R)– coil.
F7	–(R)–	&RLAT	RESET Coil	The associated discrete reference is set OFF if the coil receives power. It remains reset until set by an –(S)– coil.
F8	–(↑)–	&PCOIL	Positive Transition Coil	If the associated discrete reference is OFF when the coil receives power, the reference will be set ON for one logic scan. This coil can be used as a one-shot.
F9	–(↓)–	&NCOIL	Negative Transition Coil	If the associated discrete reference is ON and the coil is not receiving power, the reference will be set ON for one logic scan.
F10	more			Return to the first level of relay functions.

Table 3-2. Timer and Counter Functions

Key	Instruction	Mnemonic	Function	Description
Selecting Timer/Counter Functions (Shift-F2) displays these function key assignments.				
F1	ondtr	&ONDTR	On-Delay Stopwatch Timer	The ONDTR function accumulates time while receiving power. It passes power if the current value exceeds the preset value. The current value is reset to zero when the reset (R) input receives power.
F2	oftd	&OFTD	Off-Delay Timer	The OFDT function increments while power flow is off, and resets to zero when power flow is on. Time may be counted in tenths of seconds (the default selection), hundredths of seconds, or thousandths of seconds. The range is 0 to +32,767 time units. The state of this timer is retentive on power failure; no automatic initialization occurs at power-up
F3	tmr	&TMR	On-Delay Timer	The current value of the TMR function is set to zero when the function transitions on. The function accumulates time while receiving power, and passes power if the current value is greater than or equal to a preset value.
F4	upctr	&UPCTR	Up Counter	The UPCTR function increments by 1 each time the function receives transitional power. If the current value stored in the counter is greater than or equal to a preset value, the function passes power. The R input is used to reset the counter to zero.
F5	dnctr	&DNCTR	Down Counter	The DNCTR function counts down from a preset value every time the function receives transitional power. If the current value of the counter is zero, the function passes power. The R input is used to set the current value to equal the preset value.
F10	tmbase			Select the time base of a timer. Time may be counted in tenths of seconds or hundredths of seconds.
Pressing Timebase (F10) displays these function key assignments:				
F2	0.1s	_TEN	0.1 second	Time is counted in tenths of a second.
F3	0.01s	_HUN	0.01 second	Time is counted in hundredths of a second.
F4	0.001s	_TH	0.001 second	Time is counted in thousandths of a second.
F10	instrs			Return to the timer and counter functions.

Note

The Off-Delay timer and the thousandth of a second timebase are available to Release 4.5 and later of all models of CPUs, and when using Logicmaster 90-30/20 4.5 or higher. These features are not available to earlier releases of CPUs, nor earlier versions of Logicmaster.

Table 3-3. Math Functions

Key	Instruction	Mnemonic	Function	Description
Selecting Math Functions (Shift-F3) displays these function key assignments:				
F1	add	&ADD	Addition	Add two numbers. The ADD function passes power if the operation does not result in an overflow.
F2	sub	&SUB	Subtraction	Subtract one number from another. The SUB function passes power if the operation does not result in an overflow.
F3	mul	&MUL	Multiplication	Multiply two numbers. The MUL function passes power if the operation does not result in an overflow.
F4	div	&DIV	Division	Divide one number by another, yielding a quotient. The DIV function passes power if the operation does not result in an overflow and if there is no attempt to divide by zero.
F5	mod	&MOD	Modulo Division	Divide one number by another, yielding a remainder. The MOD function passes power unless there is an attempt to divide by zero.
F6	sqrt	&SQRT	Square Root	Find the square root of an integer value. When the function receives power flow, the value of the output Q is set to the square root of the input IN.
F10	types			Select a data type for the function. Pressing **Types** (**F10**) displays the function keys described in table 3-10 on page 3-12.

Table 3-4. Relational Functions

Key	Instruction	Mnemonic	Function	Description
Selecting Relational Functions (Shift-F4) displays these function key assignments:				
F1	eq	&EQ	Equal	Test for equality between two numbers. The EQ function passes power if the two inputs are equal.
F2	ne	&NE	Not Equal	Test for non-equality between two numbers. The NE function passes power if the inputs are not equal.
F3	gt	&GT	Greater Than	Test for one number greater than another. The GT function passes power if the first parameter is greater than the second parameter.
F4	ge	&GE	Greater Than or Equal To	Test for one number greater than or equal to another. The GE function passes power if the first parameter is greater than or equal to the second parameter.
F5	lt	&LT	Less Than	Test for one number less than another. The LT function passes power if the first parameter is less than the second parameter.
F6	le	&LE	Less Than or Equal To	Test for one number greater than or equal to another. The LE function passes power if the first parameter is less than or equal to the second parameter.
F7	range	&RANG	Range	Test the input value against a range of two numbers. This instruction is only available for release 4.50 or higher CPUs (4.02 of the 341) and in Logicmaster 90-30/20 Version 4.5 and above.
F10	types			Select a data type for the function. Pressing **Types** (**F10**) displays the function keys described in table 3-10 on page 3-12.

Table 3-5. Bit Operation Functions

Key	Instruction	Mnemonic	Function	Description
Selecting Bit Operation Functions (Shift-F5) displays these function key assignments:				
F1	and	&AND	Logical AND	Logical AND of two bit strings.
F2	or	&OR	Logical OR	Logical OR of two bit strings.
F3	xor	&XOR	Logical Exclusive OR	Logical Exclusive OR of two bit strings.
F4	not	&NOT	Logical Invert	Logical inversion of a bit string.
F5	shl	&SHL	Shift Left	Shift a bit string left.
F6	shr	&SHR	Shift Right	Shift a bit string right.
F7	rol	&ROL	Rotate Left	Rotate a bit string left.
F8	ror	&ROR	Rotate Right	Rotate a bit string right.
F9	more			Additional bit operation functions you can select.
F10	types			Select a data type for the function. Pressing **Types** (**F10**) displays the function keys described in table 3-10 on page 3-12.
Pressing More (F9) displays these additional bit operation function key assignments:				
F1	bittst	&BTST	Bit Test	Test a bit within a bit string.
F2	bitset	&BSET	Bit Set	Set one bit within a string to true.
F3	bitclr	&BCLR	Bit Clear	Set one bit within a string to false.
F4	bitpos	&BPOS	Bit Position	Locate a bit set to true within a bit string.
F5	mskcmp	&MCM	Masked Compare	Perform a masked compare of two arrays (available only for Release 4.5 or higher CPUs and in Logicmaster 90-30/20 Version 4.5 and above.).
F9	more			Return to the first level of bit operation functions.

Table 3-6. Data Move Functions

Key	Instruction	Mnemonic	Function	Description
Selecting Data Move Functions (Shift-F6) displays these function key assignments:				
F1	move	&MOV	Move	Move one or more bits of data within PLC memory.
F2	blkmov	&BLKMOV	Block Move	Move a block of up to 7 constants to PLC memory.
F3	blkclr	&BLKCLR	Block Clear	Clear (0) one or more bytes/words of PLC memory.
F4	shfreg	&SHFR	Shift Register	Shift one or more words or bits of data through a block of PLC memory.
F5	bitseq	&BITSEQ	Bit Sequencer	Sequence a 1 through a group of bits in PLC memory.
F7	comreq	&COMMREQ	Communication Request	Send a communications request to a smart module in the PLC.
F10	types			Select a data type for the function. Pressing **Types** (**F10**) displays the function keys described in table 3-10 on page 3-12.

Table 3-7. Table Functions

Key	Instruction	Mnemonic	Function	Description
Selecting Table Functions (Shift-F7) displays these function key assignments:				
F1	srh eq	&SRCHEQ	Search Equal	Search array for values equal to a specified value.
F2	srh ne	&SRCHNE	Search Not Equal	Search array for values not equal to a specified value.
F3	srh gt	&SRCHGT	Search Greater Than	Search array for values greater than a specified value.
F4	srh ge	&SRCHGE	Search Greater Than or Equal	Search array for values greater than or equal to a specified value.
F5	srh lt	&SRCHLT	Search Less Than	Search array for values less than a specified value.
F6	srh le	&SRCHLE	Search Less Than or Equal	Search array for values less than or equal to a specified value.
F8	arrmov	&ARRMOV	Array Move	Copy a specified number of data elements from a source array to a destination array.
F10	types			Select a data type for the function. Pressing **Types** (**F10**) displays the function keys described in table 3-10 on page 3-12.

Table 3-8. Conversion Functions

Key	Instruction	Mnemonic	Function	Description
Selecting Conversion Functions (Shift-F8) displays these function key assignments:				
F3	→bcd-4	&TO_BCD4	Convert to BCD-4	Convert a value to 4-digit BCD format. The →BCD-4 function passes power unless the number to be converted is out of range (greater than 9999), and no conversion is performed.
F6	→int	&TO_INT	Convert to INT	Convert a value to signed integer format. The →INT function passes power unless the number to be converted is out of range (-32,768 to +32,767), and no conversion is performed.
F10	types			Select the type of data for the function. Pressing **Types** (**F10**) displays the function keys described in table 3-10 on page 3-12.

Table 3-9. Control Functions

Key	Instruction	Mnemonic	Function	Description
Selecting Control Functions (Shift-F9) displays these function key assignments:				
F1	call	&CALL	Call	Cause a program execution to go to a specified subroutine block. The CALL function always passes power.
F2	do io	&DOIO	Do I/O	Service a specified range of inputs or outputs immediately (all inputs or outputs on a module will be serviced if any addresses on that module are included in the function – partial I/O module updates are not performed). Optionally, a copy of the scanned I/O can be placed in internal memory.
F4	pidisa	&PIDISA	ISA PID Algorithm	Select the standard IDS PID algorithm.
F5	pidind	&PIDIND	Independent PID Algorithm	Select the non-interacting independent PID algorithm.
F7	end	&END	Temporary End of Logic	The program executes from the first rung to the last rung or the END instruction, whichever is encountered first. This instruction is useful for debugging purposes.
F8	commnt	&COMMENT	Comment	A rung explanation. After programming the instruction, the text can be typed in by zooming into the instruction.
F9	svcreq	&SVCREQ	Service Request	A special PLC service function. This function passes power if power is received and the function executes properly.
F10	more			Additional control functions you can select.

Table 3-9. Control Functions (cont'd)

Key	Instruction	Mnemonic	Function	Description
Pressing More (F10) displays these additional control function key assignments:				
F1	mcrn	&MCRN	Nested Master Control Relay	Start a master control relay range. This is the nested form of the MCR instruction. An MCRN causes all rungs between the nested MCRN and its subsequent ENDMCRN to be executed with no power flow. There can be nothing after an MCRN in a rung.
F2	endmcn	&ENDMCRN	Nested End Master Control Relay	End a nested master control relay range. This is the nested form of the ENDMCR instruction. There can be nothing after a nested MCR in a rung.
F3	jumpn	&JUMPN	Nested Jump	Jump to a specified location indicated by a LABELN in the logic. This is the nested form of the JUMP instruction.
F4	labeln	&LABELN	Nested Label	The target location of a JUMP instruction. This is the nested form of the LABEL instruction.
F6	mcr	&MCR	Non-Nested Master Control Relay	Start a non-nested master control relay range. This is the non-nested form of the MCR instruction. A non-nested MCR causes all rungs between the non-nested MCR and its subsequent non-nested ENDMCR to be executed with no power flow. There can be nothing after a non-nested MCR in a rung.
F7	endmcr	&ENDMCR	Non-Nested End Master Control Relay	End a non-nested Master Control Relay range. This is the non-nested form of the ENDMCR instruction. There can be nothing after a non-nested ENDMCR in the rung.
F8	jump	&JUMP	Non-Nested Jump	Jump to a specified location indicated by a LABEL in the logic. This is the non-nested form of the JUMP instruction.
F9	label	&LABEL	Non-Nested Label	The target location of a JUMP instruction. This is the non-nested form of the LABEL instruction.
F10	more			Return to the first level of control functions.

Table 3-10. Data Types

Key	Instruction	Mnemonic	Function	Description
Selecting Data Types (F10) displays these function key assignments:				
F1	bit	_BI	Bit	A Bit data type is used with instructions which operate on bit strings that are not multiples of 16 bits, or whose reference address is not on a byte boundary (e.g., MOV or SHFREG).
F2	byte	_BY	Byte	A Byte has an 8-bit value.
F3	word	_W	Word	A Word data type uses 16 consecutive bits of data memory; but, instead of the bits in the data location representing a number, the bits are independent of each other. Each bit represents its own binary state (1 or 0), and the bits are not looked at together to represent an integer number. The valid range of word values is 0 to +65,535.
F5	bcd-4	_BCD4	Four-Digit Binary Coded Decimal	Four-digit BCD numbers use 16-bit data memory locations. Each BCD digit uses four bits and can represent numbers between 0 and 9. This BCD coding of the 16 bits has a legal value range of 0 to 9999.
F8	int	_INT	Signed Integer	Signed integers use 16-bit memory data locations, and are represented in 2's complement notation. The valid range of an INT data type is -32,768 to +32,767.
F9	dint	_DI	Double Precision Integer	Double precision integers are stored in 32-bit data memory locations (actually two consecutive 16-bit memory locations) and are always signed values. (Bit 32 is the sign bit.) The valid range of a DINT data type is -2147483648 to +2147483867.
F10	instrs			Return to the functions displayed on the screen.

Data Zoom

The data zoom feature supports the PID functions by providing a display window that expands the parameters of the function block and displays them in real time with labels in a format consistent with their use. For example, a word of data may contain several boolean flags. Each flag is labeled and displayed separately. The boolean inputs and outputs to the function are also displayed.

The data zoom feature is available in either **ONLINE** or **OFFLINE** mode. In **ONLINE** mode when the program folder is identical to the PLC, changes made to values are only stored to the PLC. Real-time updates are maintained within the data zoom window. In **OFFLINE** mode, changes made to values are only stored to disk.

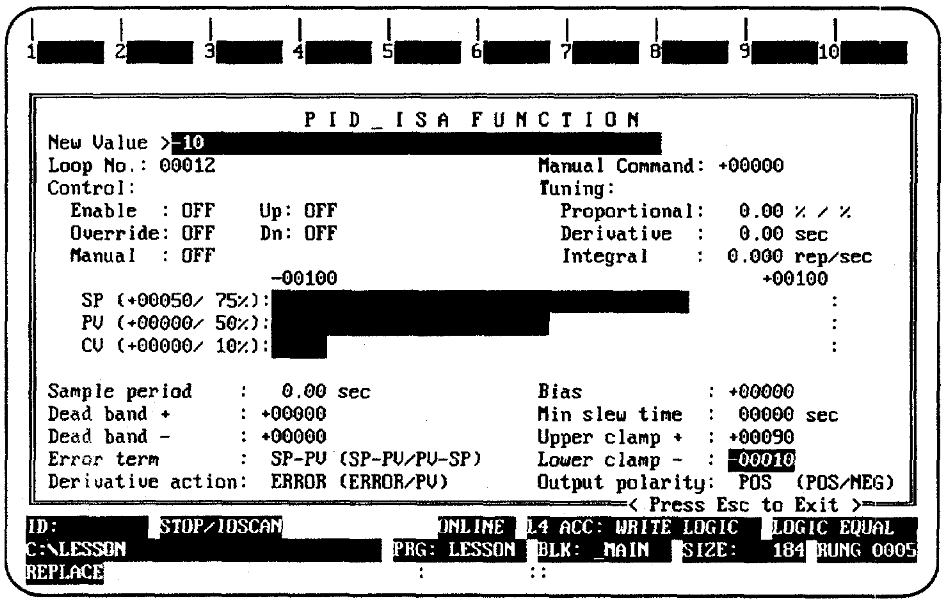
Note

If you try to write-protect a floppy disk while the Data Zoom screen is displayed, any changes made on the Data Zoom screen will be lost when you exit the screen.

Using the Data Zoom Feature

The data zoom feature is only available in the program editor. To use this feature:

1. Position the cursor within a PID function block and press **Zoom** (**F10**). The following screen shows an example screen for the PID_ISA function.



```
 1     2     3     4     5     6     7     8     9     10

                  P I D _ I S A   F U N C T I O N
 New Value >-10
 Loop No.: 00012                             Manual Command: +00000
 Control:                                    Tuning:
   Enable  : OFF    Up: OFF                    Proportional:   0.00 % / %
   Override: OFF    Dn: OFF                    Derivative  :   0.00 sec
   Manual  : OFF                               Integral    :  0.000 rep/sec
                   -00100                                       +00100
    SP (+00050/ 75%):                                                :
    PV (+00000/ 50%):                                                :
    CV (+00000/ 10%):                                                :

 Sample period     :   0.00 sec              Bias            : +00000
 Dead band +       : +00000                  Min slew time   :  00000 sec
 Dead band -       : +00000                  Upper clamp +   : +00090
 Error term        :  SP-PV (SP-PV/PV-SP)    Lower clamp -   : -00010
 Derivative action:  ERROR (ERROR/PV)        Output polarity:  POS  (POS/NEG)
                                                    < Press Esc to Exit >
ID:          STOP/IOSCAN              ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LESSON                       PRG: LESSON  BLK: _MAIN  SIZE:   184 RUNG 0005
REPLACE                                   :         ::
```

2. The PID function is displayed as a window, with the cursor positioned on the first field whose value can be changed.

   Each field in the window is displayed in a format consistent with its usage in the function. In the example above, *Loop No.* indicates the number this PID is within a loop structure and is, therefore, displayed as a signed integer; and *Min slew time* is a timing parameter displayed in seconds. Fixed point numbers, like those displayed in the *Sample period* field, are truncated upon entry.

While the data zoom window is active, the function softkeys normally displayed at the top of the screen will be blank.

3. The *New Value* field functions like a command line. To change the value of any field (provided that field can be changed), use the arrow keys to highlight the field, enter a new value in the *New Value* field, and press the **`Enter`** key. The value of the field will be changed when the **`Enter`** key is pressed. You can also use the **`Tab`** key to change the value of a field.

4. Most field values associated with explicit parameters can be changed. The software will not allow you to highlight those fields whose values cannot be changed.

   In the PID function, the *CV, PV,* and *Enable* fields cannot be changed. The following table lists the formulas used to compute the percentages in these fields.

Parameter	Formula
SP	Percentage = (quantity - min value) / (max value - min value), where quantity is the current value of SP. Using this formula with the values displayed on the Data Zoom screen shown on the previous page results in a value of 75% for SP: SP = [50 - (-100)] / [+100 - (-100)] = 150 / 200 = .75 or 75%
PV	Percentage = (quantity - min value) / (max value - min value), where quantity is the current value of PV. Using this formula with the values displayed on the Data Zoom screen shown on the previous page results in a value of 50% for PV: PV = [0 - (-100)] / [+100 - (-100)] = 100 / 200 = .50 or 50%
CV	Percentage = (CV - lower clamp) / (upper clamp - lower clamp). Using this formula with the values displayed on the Data Zoom screen shown on the previous page results in a value of 10% for CV: CV = [0 - (-10)] / [+90 - (-10)] = 10 / 100 = .10 or 10%

   Percentages are calculated to the nearest 1% on the screen. Percentages less than zero are set to 0%. Percentages greater than 999 are set to 999%. The bar will display up to 100%.

   Whenever a bar graph is displayed in data zoom, minimum and maximum scaling values will also need to be displayed. For the PID function, the recommended default minimum and maximum values for the SP and PV bars are -32,000 and +32,000. The minimum and maximum scaling values for the CV bar are the lower and upper clamp values. In the screen shown above, the −100 and +100 displayed immediately above the bar graphs correspond to the minimum and maximum SP and PV scaling values.

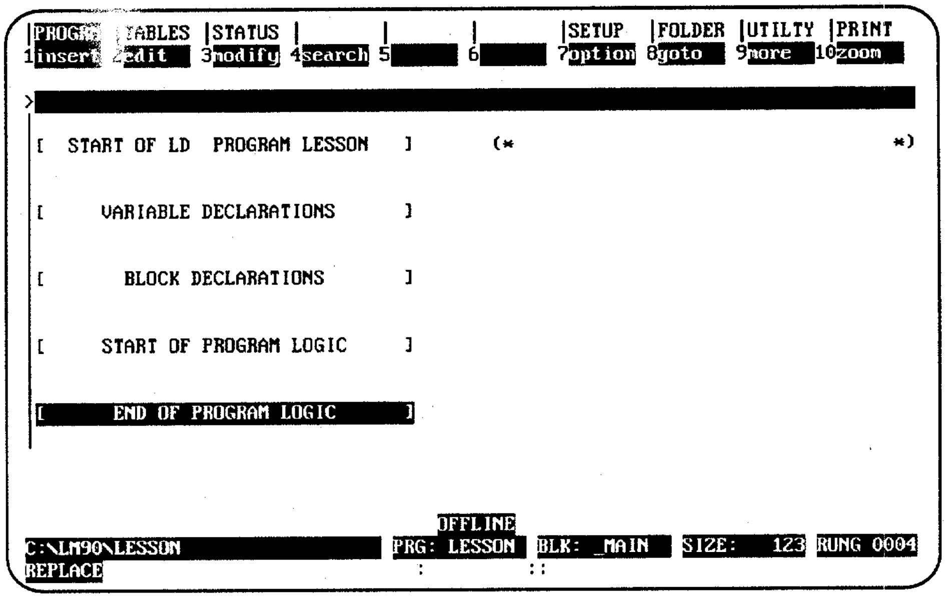
5. Press the **`Escape`** key to exit the data zoom window and return to your original position within the editor before the data zoom feature was begun.

For more information on the PID function and its parameters, refer to the *Series 90-30/20/Micro Programmable Controllers Reference Manual,* GFK-0467.

Section 2: Program Format

Program elements are combined to form rungs of ladder logic. A ladder diagram has a symbolic power source. Power is considered to flow from the left rail through a contact to the coil or function block connected to the right.

From the main menu, select **Program Display/Edit (`F1`)**. The screen displays a list of markers which represent parts of a program.



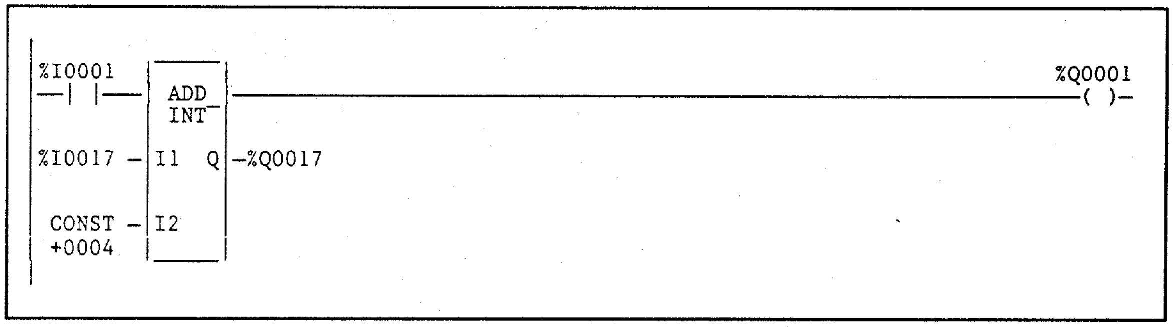
```
|PROGR  TABLES |STATUS |        |        |        |SETUP   |FOLDER  |UTILTY  |PRINT
1insert 2edit   3modify 4search 5        6        7option 8goto    9more   10zoom
>

[  START OF LD  PROGRAM LESSON   ]        (*                                   *)

[     VARIABLE DECLARATIONS      ]

[       BLOCK DECLARATIONS       ]

[     START OF PROGRAM LOGIC     ]

[       END OF PROGRAM LOGIC     ]


                                     OFFLINE
C:\LM90\LESSON                       PRG: LESSON  BLK: _MAIN  SIZE:   123 RUNG 0004
REPLACE                                  :          : :
```

Marker	Description
Variable Declarations	To access the variable declaration table, move the cursor to this marker and press **`Zoom`** (**`F10`**). Nicknames and reference descriptions can then be entered in the table.
Block Declarations	A program can include more than one block of logic. Additional blocks, known as subroutine blocks, can be called from other blocks. When that is done, blocks must be declared before they are called. The main block has a block declaration table. This table lists all blocks which are part of the complete program. Blocks do not have block declaration tables. However, blocks can be called from the main block or from any block in the program.
Start/End of Program Logic	All logic is placed between these two markers. To enter logic, place the cursor on the **`[ END OF PROGRAM LOGIC ]`** marker and press **`Insert`** (**`F1`**).

The cursor keys are used to highlight the area of the program to be displayed or edited.

Creating or Editing Program Logic

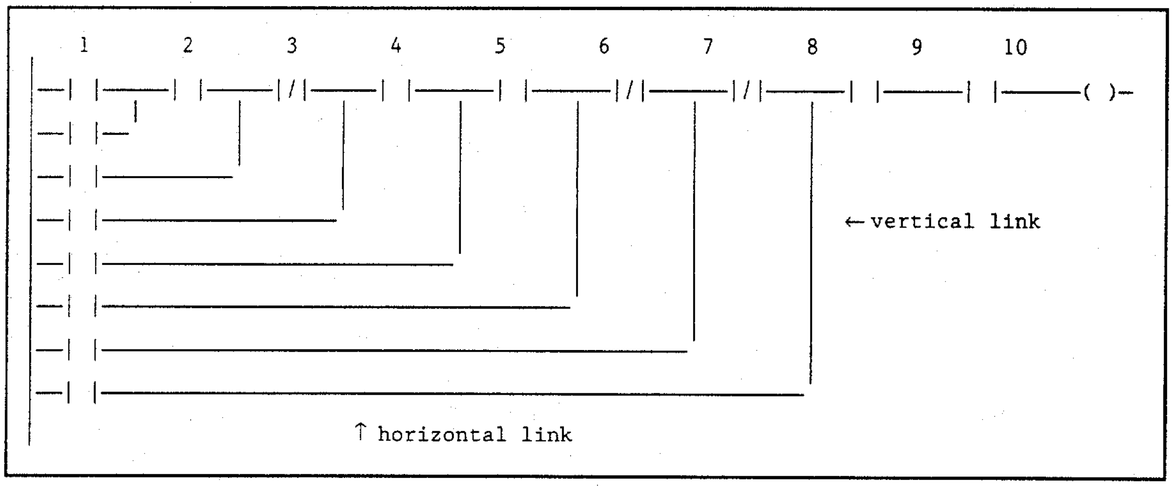
Program logic consists of various elements such as relays, timers, math functions, and other functions, placed together to form rungs of logic.



Structure of a Ladder Logic Rung

The programmer allows great flexibility in entering program elements; however, it will not allow you to enter a rung with incorrect format or syntax.

Each rung may contain up to eight parallel lines; each line may have up to ten elements connected in series. Examples of an element include a normally open contact, a normally closed contact, or a coil. Horizontal and vertical links are used to carry power around an element, or to place elements in parallel or series with one another.



Note

Programs created using the Hand-Held Programmer must conform to this format to be totally compatible with Logicmaster 90-30/20/Micro software.

The following example shows two separate rungs, which must be entered and accepted separately.

```
|
|—| |——| |——| |————————————————————————————————————————( )—
|—| |——| |——|/|————————————————————————————————————————( )—
|
```

In the next example, two rung lines are connected by a vertical link, forming only one rung.

```
|
|—| |—+—| |——| |———————————————————————————————————————( )—
|      |
|      +—| |——|/|———————————————————————————————————————( )—
|
```

The last element of a group of rung elements in series must be a coil, a jump, or a function. Nothing may be to the right of a coil or a jump. The tenth position of a rung line is reserved for coils and jumps. A call instruction may occupy columns 9 and 10. A rung may contain up to eight coils.

A rung line is not required to have elements in each column.

Ladder Logic Language Rules

These guidelines should be followed when creating or editing ladder logic:

1. If a rung has a transitional coil, it must be the only coil in the rung.
2. There can be only one JUMP or MCR per rung. It must be the last instruction in the rung, and there cannot be a coil in the same rung.
3. A rung must contain at least one contact before any coil, jump, MCR, function, or vertical link. Contacts ***must*** be entered and cannot be left blank. Function blocks cannot be tied directly into the power rail.
4. Short circuits are not allowed.

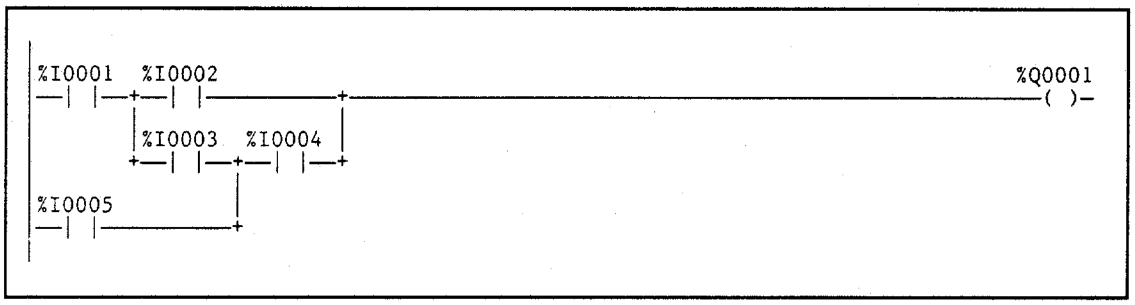
Note

The ALW_ON contact, shown below, may be used to satisfy rules 3 and 4 above.

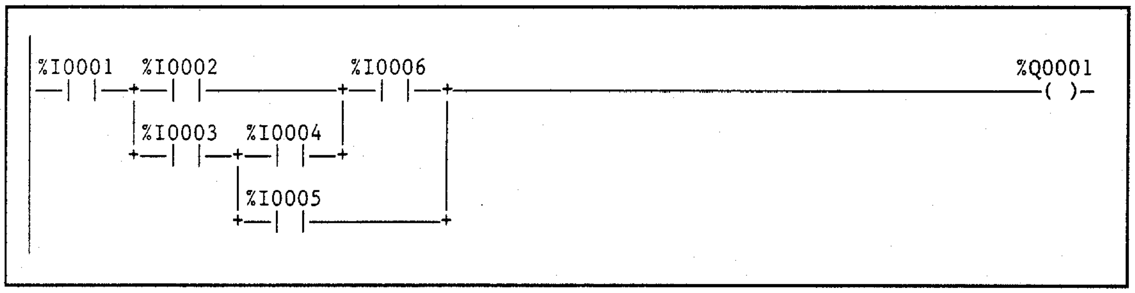
```
ALW_ON
—|¯|—
```

5. A rung must be composed of properly nested sub-expressions. There can be no branches either into or out of another branch. The following examples contain improperly nested rungs.

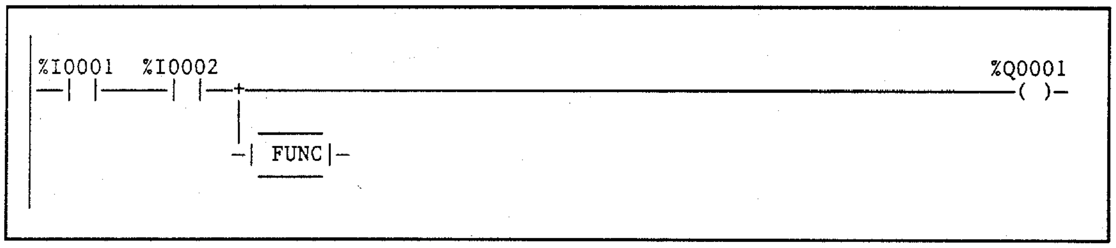
   A. In this example, the rung line containing the %I0005 contact branches into the middle of the sub-expression (%I0002 OR (%I0003 AND %I0004)).



   B. In this example, the rung line containing the %I0005 contact branches out of the middle of the sub-expression (%I0002 OR (%I0003 AND %I0004)).



6. There can be no branch around (above or below) a function in a rung. The following rung is not allowed.



7. There can be no sub-paths starting from a vertical in a rung containing a function, except for sub-paths leading directly to coils.

   A. The following rung is allowed because the first sub-path comes directly off the power rail and the second leads directly to coils.

```
|
|%I0001  %I0002          |-----|                                   %Q0001
|--| |-----| |-----------| FUNC|------------------------------+---------( )-
|                        |  X  |                              |
|%I0001  %I0002  %I0003  |     |                              |     %Q0002
|--| |-----| |-----| |---|     |                              +---------( )-
|                        |_____|
|
```

   B. The next rung is not allowed. It has a sub-path starting from a vertical and leading into the function. It also has a sub-path that does not lead directly to coils; it goes through contacts first.

```
|
|%I0001  %I0002          |-----|                        %I0004     %Q0001
|--| |-----| |--+--------| FUNC|-----------------------+--| |---------( )-
|               |        |  X  |                       |
|               |%I0003  |     |                       |%I0005     %Q0002
|               +--| |---|     |                       +--| |---------( )-
|                        |_____|
|
```

8. There can be no contacts following a function in a rung. Note that the rung in the last example above fails this rule, too.

9. In general, execution order of rung elements is left-to-right. Within a group of parallel branches, the first (lowest rung line) parallel branch is executed first. The first of multiple sub-paths is executed first.

This page is intentionally left blank.

Section 3: Program Entry

Logicmaster 90-30/20/Micro software was designed to allow for rapid entry of relay ladder diagram programs. By allowing entry of program elements with either function keys or mnemonics, both frequent and occasional users can be satisfied.

Annotation (nicknames, reference descriptions, and rung comments) can be input either prior to logic entry or as each logic element is entered. Rung comments can be entered as logic is created or inserted after the logic has been debugged. For information on annotation, refer to chapter 3, section 4, "Program Annotation."

Note

Program folders on write-protected floppy disks are automatically locked. Remove the write-protect tab and unlock the folder using the lock/unlock current program folder function (see chapter 7).

Using Mnemonics

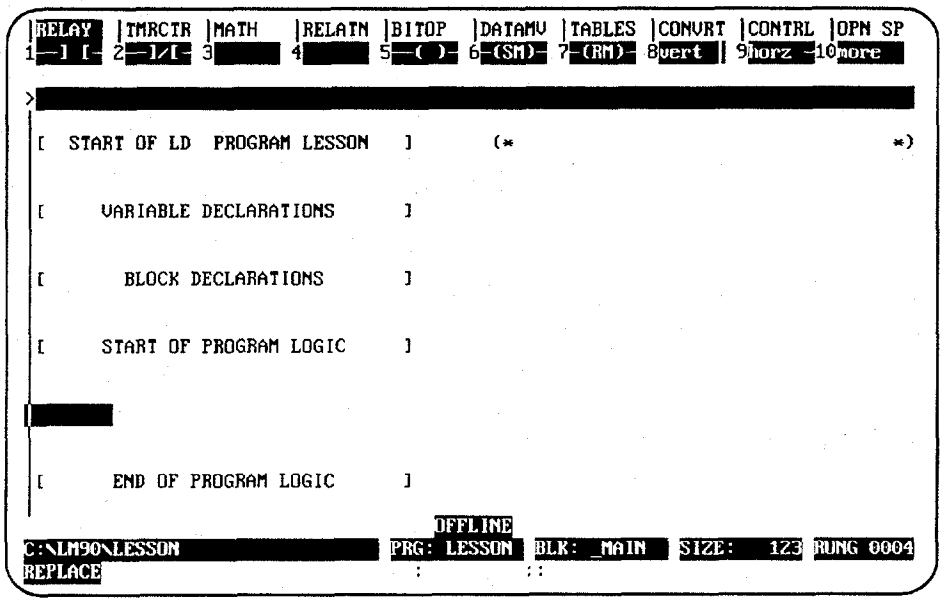
Mnemonic entry enables you to enter an instruction by typing its mnemonic on the command line. For example, to enter the ADD function, you would type **`%ADD`** on the command line and press the **`Enter`** key.

For some instructions, it is not necessary to type the entire mnemonic, just enough characters for the entry to be unique. For example, instead of typing **`&ADD`** to enter the ADD function, you could simply type **`&AD`** and press the **`Enter`** key. Appendix D, *Instruction Mnemonics,* lists the mnemonics of all program instructions. While programming, you can also display a list of mnemonics by pressing **`ALT-I`**.

For many functions, you can also specify a data type or reference address. For example, **`&ADD_DINT`** would enter the double precision integer version of the ADD function at the current cursor position in the rung.

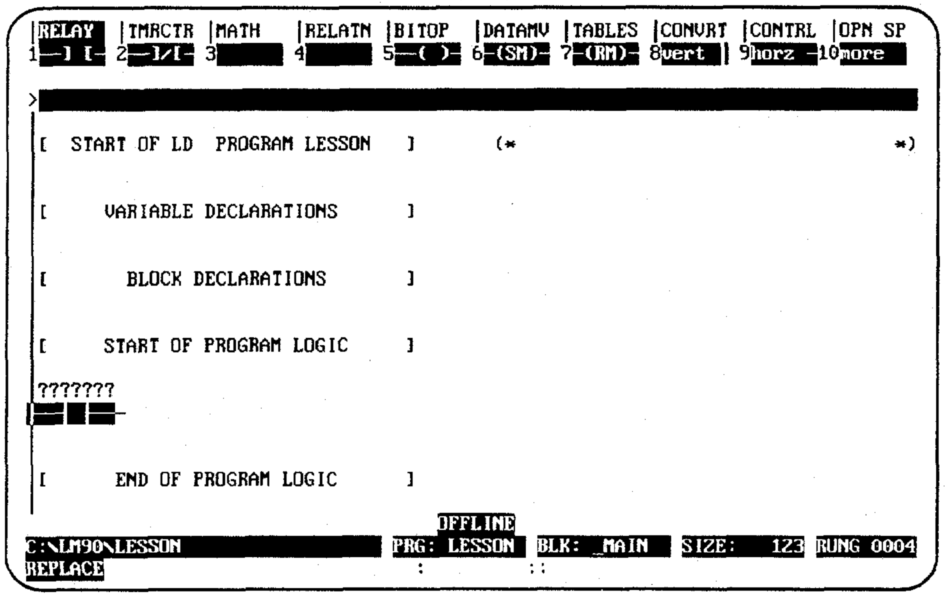
Inserting Logic Elements

1. With the cursor on the **[END OF PROGRAM LOGIC]** marker, press **Insert** (**F1**). Rungs are always inserted before the rung the cursor is on.



```
|RELAY  |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1-] [- 2-]/[- 3       4       5-( )- 6-(SM)- 7-(RM)- 8vert | 9horz -10more
>
[  START OF LD  PROGRAM LESSON    ]      (*                              *)

[     VARIABLE DECLARATIONS       ]

[       BLOCK DECLARATIONS        ]

[     START OF PROGRAM LOGIC      ]



[      END OF PROGRAM LOGIC       ]
                                       OFFLINE
C:\LM90\LESSON                    PRG: LESSON  BLK: _MAIN  SIZE:    123 RUNG 0004
REPLACE                                :          ::
```

2. To enter a relay element at the cursor location, press the desired function key. For example, to enter a normally open contact, press **F1** with the relay functions displayed for the softkeys at the top of the screen. Or, you may enter the normally open contact by typing the mnemonic **&NOCON** on the command line and pressing the **Enter** key.



```
|RELAY  |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1-] [- 2-]/[- 3       4       5-( )- 6-(SM)- 7-(RM)- 8vert | 9horz -10more
>
[  START OF LD  PROGRAM LESSON    ]      (*                              *)

[     VARIABLE DECLARATIONS       ]

[       BLOCK DECLARATIONS        ]

[     START OF PROGRAM LOGIC      ]
???????
|--] [--

[      END OF PROGRAM LOGIC       ]
                                       OFFLINE
C:\LM90\LESSON                    PRG: LESSON  BLK: _MAIN  SIZE:    123 RUNG 0004
REPLACE                                :          ::
```

3. A reference may now be entered on the command line. For each reference, include both a user reference and location. There are two ways to do this:

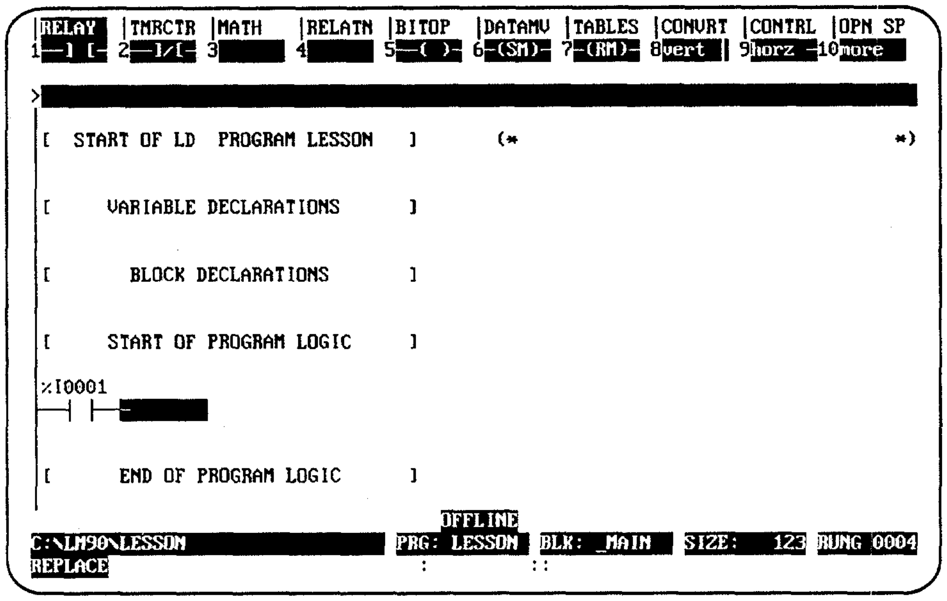
   A. By entering the reference type and then the address (e.g., %I1), or

   B. By entering them in reverse order (e.g., 1I). The software automatically places the entry in the correct order and format when you press the **Enter** key. Entering the number before the reference type eliminates having to enter the % character.

   Then, press the **Enter** key. The cursor automatically advances to the next position, ready for entry of the next element.

   Some program functions require references that begin on a word or byte boundary. The Logicmaster 90-30/20/Micro software will automatically adjust the entries to be properly aligned.

   You may also combine the previous step and this step into one operation by typing the mnemonic **&NOCON I1** on the command line and pressing the **Enter** key.

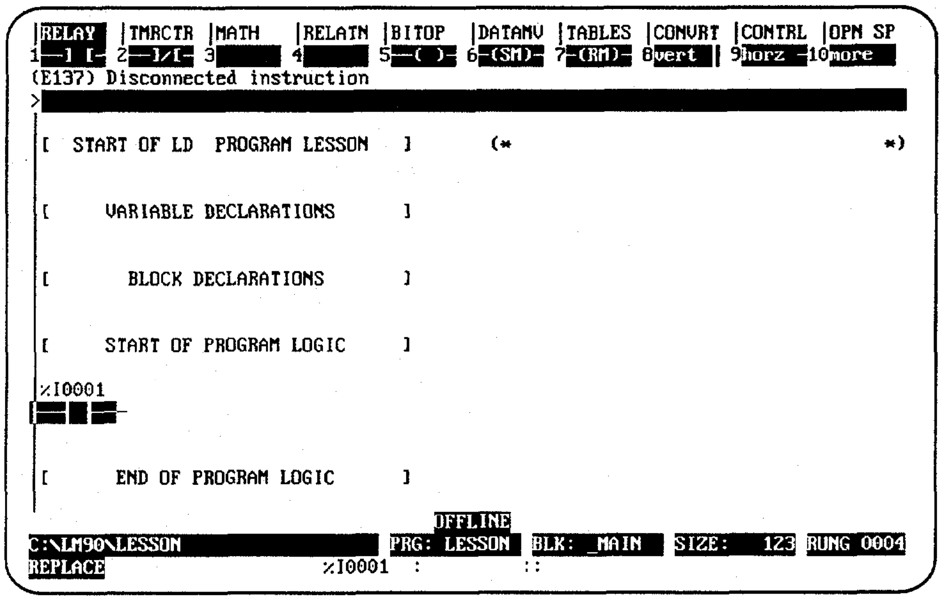


```
|RELAY |TMRCTR |MATH  |RELATN |BITOP |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1-] [- 2-]/[- 3      4       5-( )- 6-(SM)- 7-(RM)- 8vert  9horz -10more
>
[  START OF LD  PROGRAM LESSON   ]      (*                               *)

[    VARIABLE DECLARATIONS       ]

[      BLOCK DECLARATIONS        ]

[    START OF PROGRAM LOGIC      ]

%I0001
--] [--

[     END OF PROGRAM LOGIC       ]
                                  OFFLINE
C:\LM90\LESSON                   PRG: LESSON  BLK: _MAIN  SIZE:   123 RUNG 0004
REPLACE                              :          ::
```

Note

The previous two operations can be combined into one by entering the reference address before pressing the contact function key.

4. This process can be continued until a rung is completed and is ready to be accepted. A rung can be accepted by pressing the **Enter** key with the command line empty, or the Plus (+) key on the numeric keypad. If there is an error in the rung, the rung is not accepted and the cursor is placed on the incorrect element for correction.



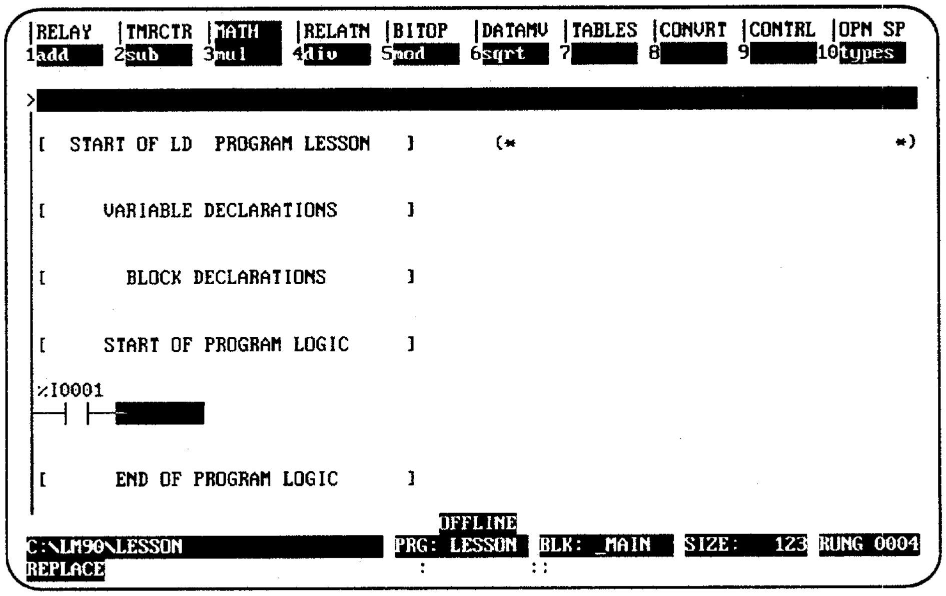
```
|RELAY  |TMRCTR |MATH    |RELATN |BITOP   |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1-] [-  2-]/[-  3       4       5-( )-   6-(SM)- 7-(RM)- 8vert |  9horz - 10more
(E137) Disconnected instruction
>

[  START OF LD  PROGRAM LESSON   ]       (*                                    *)

[     VARIABLE DECLARATIONS      ]

[       BLOCK DECLARATIONS       ]

[     START OF PROGRAM LOGIC     ]

%I0001


[      END OF PROGRAM LOGIC      ]
                                          OFFLINE
C:\LM90\LESSON                           PRG: LESSON  BLK: _MAIN   SIZE:    123 RUNG 0004
REPLACE                        %I0001  :           ::
```

Inserting Functions

Functions can be entered as easily as relay elements.

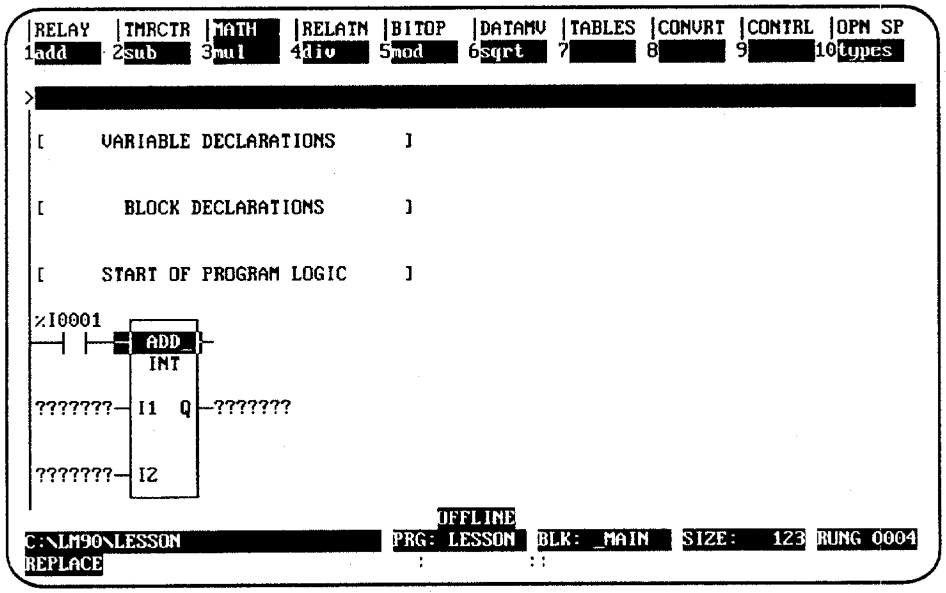
1. First, use the relay function keys to enter the enabling logic. In the first example screen shown below, a normally open contact with reference address %I1 is entered at the enabling logic.

2. Select the type of function using the shift-function keys. For example, to select math functions, press **Shift-F3**.



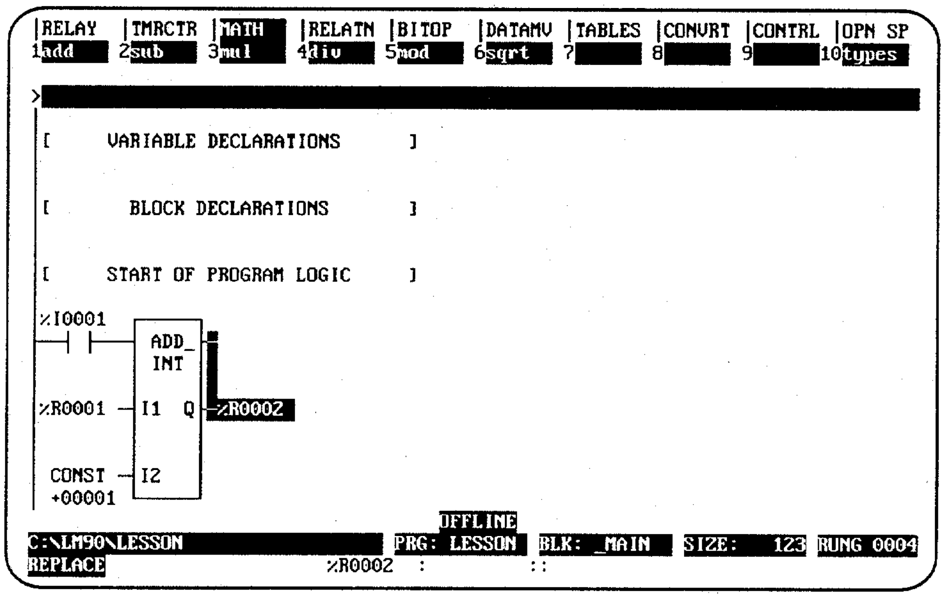
```
|RELAY  |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1add    2sub    3mul    4div    5mod    6sqrt   7       8       9       10types
>
[  START OF LD  PROGRAM LESSON   ]       (*                                  *)

[     VARIABLE DECLARATIONS      ]

[       BLOCK DECLARATIONS       ]

[     START OF PROGRAM LOGIC     ]
%I0001
--| |--

[      END OF PROGRAM LOGIC      ]
                                   OFFLINE
C:\LM90\LESSON                   PRG: LESSON  BLK: _MAIN   SIZE:    123 RUNG 0004
REPLACE                             :            ::
```

3. Select the function desired using the function keys. For example, to select the ADD function, press **Add** (**F1**). Or, you may enter the ADD function by typing the mnemonic: **&ADD** on the command line and pressing the **Enter** key.



```
|RELAY  |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1add    2sub    3mul    4div    5mod    6sqrt   7       8       9       10types
>
[     VARIABLE DECLARATIONS      ]

[       BLOCK DECLARATIONS       ]

[     START OF PROGRAM LOGIC     ]
%I0001
--| |--+ ADD_ +-
       |  INT |
???????-| I1  Q|-???????
       |      |
???????-| I2   |
                                   OFFLINE
C:\LM90\LESSON                   PRG: LESSON  BLK: _MAIN   SIZE:    123 RUNG 0004
REPLACE                             :            ::
```

4. The Tab key can now be used to move the cursor to each parameter position around the function.
5. When the cursor is on the first input, simply type the value to be entered in this position on the command line (for example, %R0001 or 1R), and press the **Tab** key or Enter key to place this entry into its position. Pressing the **Tab** key will move the cursor to the next entry position. This process can be continued until the rung is completed and accepted.



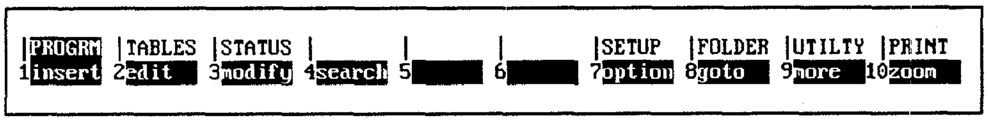
```
|RELAY  |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1add    2sub    3mul    4div    5mod    6sqrt   7       8       9       10types
>
[     VARIABLE DECLARATIONS      ]

[       BLOCK DECLARATIONS       ]

[     START OF PROGRAM LOGIC     ]

%I0001
--| |--ADD_
       INT
%R0001 -I1  Q-%R0002

CONST -I2
+00001
                                      OFFLINE
C:\LM90\LESSON                        PRG: LESSON  BLK: _MAIN  SIZE:   123  RUNG 0004
REPLACE                    %R0002  :          ::
```

For information on entering comments, refer to chapter 3, section 6, "Rung Comments." For information on zooming into entries, refer to chapter 3, section 1, "Ladder Logic Program Elements."

Exiting Rung Entry

There are three ways to exit from a rung.

1. Press the **Escape** key to attempt to accept the current rung. If the rung passes the software validity test, the original function key selections (shown below) are restored and the new logic is added to the program. After accepting a rung, the cursor moves to the next rung, ready for the next rung to be entered. However, in **INSERT** mode a new rung is automatically opened below the newly created rung.



If the logic fails the check, an error message is displayed and the cursor is positioned on the location where the error occurred.

Informational messages relating to the executability of the logic may also be displayed.

2. Press the **Enter** key with the command line empty (or the **Plus (+)** key on the numeric keypad). This causes the same response as pressing the **Escape** key.

3. Press **ALT-A** to exit the insert or edit function without modifying the existing ladder diagram logic. Confirmation is required. Pressing **ALT-A** the first time will restore the rung to its original state. In **INSERT** mode, this would be an empty rung. Pressing **ALT-A** a second time exits the insert function.

Each rung is not saved to disk as it is accepted. To update the disk, press the **Escape** key to exit the program editor and return to the Programming Software main menu, or press **ALT-U**.

Using the Cursor to Select a Reference Table

You may go directly to the reference table of a reference (e.g., of a function block operand, contact, coil, etc.) under the cursor by pressing **ALT-F2**. Then, press **Shift-F1** from the reference table to return to the same place in the program.

This page is intentionally left blank.

Section 4: Program Annotation

Annotation is optional explanatory text in a program. This text makes the program easier to read and to understand.

Note

Files containing program annotation exist only in the folder, not in the PLC.

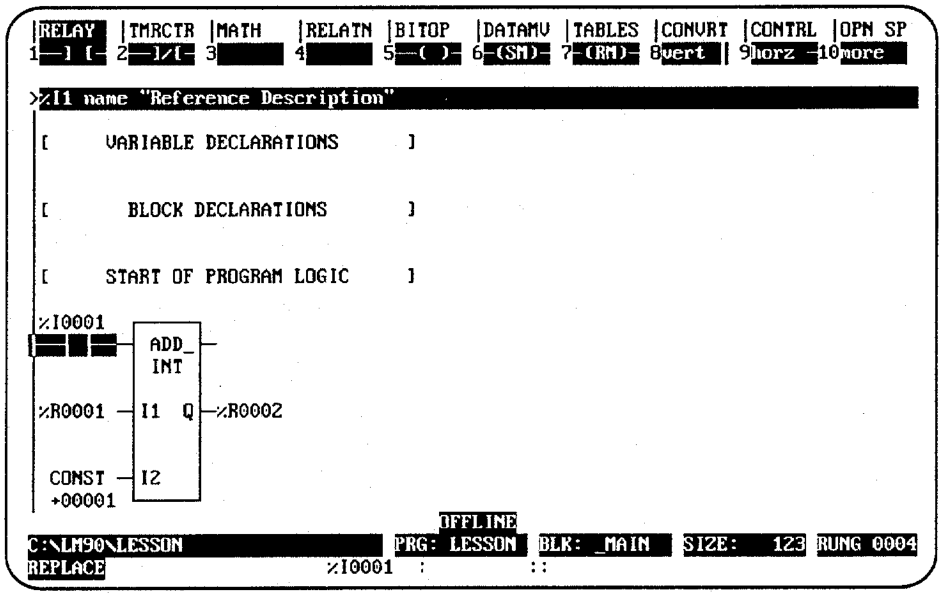
Logicmaster 90-30/20/Micro software provides the following types of program annotation:

Table 3-11. Program Annotation

Type	Description
Nickname	An optional 1- to 7-character identifier, which can be used for each program reference. The characters used in a nickname may be a letter from A through Z, a numeral from 0 through 9, an underscore, or the special characters +, -, %, #, @, <, >, =, and &. The first character of the nickname must be a letter. A nickname is case-insensitive, unless a global nickname has been reassigned in a subroutine. When a global nickname (e.g., SWITCH1) has been reassigned in a subroutine, the local use of that nickname will remain in upper-case letters; however, the global use of that nickname will be displayed in lower-case letters (e.g., switch1).
Reference Description	An optional text description of up to 32 characters which is associated with a machine reference or with implicit identifiers (e.g.,program name, subroutine block, or JUMP/LABEL/MCR/ENDMCR). A reference description can be used with or without a nickname.
Comment	Longer blocks of text (rung explanations). A comment consists of up to 2048 characters of text. On the screen, the text of a comment can be read by pressing **Zoom** (**F10**) with the cursor located at the comment rung. The comment can also be printed as part of the ladder logic.

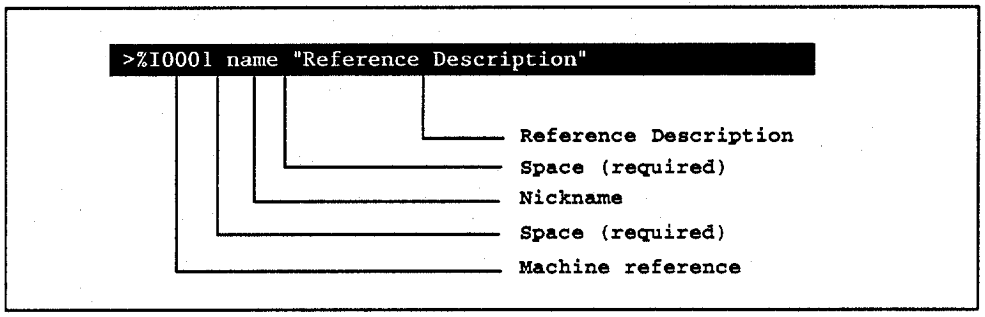
Entering Nicknames and Reference Descriptions

Nicknames and reference descriptions can be entered in two different ways. The first way is to create a nickname while programming. This is done by entering the nickname and/or reference description on the command line as the reference is used.



```
|RELAY |TMRCTR |MATH  |RELATN |BITOP |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1-] [- 2-]/[- 3      4       5-( )- 6-(SM)- 7-(RM)- 8vert | 9horz -10more

>%I1 name "Reference Description"

[    VARIABLE DECLARATIONS      ]

[      BLOCK DECLARATIONS       ]

[    START OF PROGRAM LOGIC     ]

%I0001
      -| ADD_ |-
       | INT  |

%R0001 -| I1  Q|-%R0002

 CONST -| I2   |
 +00001

                                          OFFLINE
C:\LM90\LESSON                           PRG: LESSON  BLK: _MAIN  SIZE:    123 RUNG 0004
REPLACE                        %I0001  :          ::
```

The order of entry is not important. For example:



```
>%I0001 name "Reference Description"
```

Separate the reference, nickname, and reference description by a space. Use quotation marks before and after the reference description. A double colon (::) may also be inserted between the nickname and the reference description. If a double colon is used, quotation marks are not required around the reference description.

When you press the **Enter** key, if there are no conflicts, the nickname is displayed above the program element. It is also temporarily included in the variable declaration table for that program. While inserting or editing a rung, you can press **ALT-V** to view the variable declaration table. Temporary entries are marked with an asterisk. When the rung is accepted, the nickname is permanently added to the table. If the rung is aborted before it is accepted, this information is removed from the table.

If a conflict or error occurs when you press the **Enter** key, the contents of the command line are not applied to the instruction and a message is displayed. Press **CTRL-Home** to recall the last entry made on the command line. Then, press the **CTRL** key and the Left or Right cursor key to move within the command line to correct the entry.

Depending on which display mode is active, if you enter a nickname for a reference, it is displayed in the program instead of the reference. For example, if you entered the reference %I0104 for a normally closed contact in the logic, the display would look like this:

```
|
|%I0104
|—|/|—
|
```

If you entered a nickname and (optionally) a reference description for the reference on the command line:

```
%I0104 XWATMOV "'x' APM waiting move"
```

The display would look like this instead:

```
|
|XWATMOV
|—|/|—
|
```

Note

Use **ALT-N** to toggle between reference address display, nickname display, reference description display, and compressed rung display. Refer to chapter 6, "Programmer Setup," for more information on specifying which modes will be displayed when **ALT-N** is pressed.

The second and simplest way to enter nicknames and reference descriptions, however, is to use the variable declaration table, described in the next section. As each entry is made in the variable declaration table, the reference description for the current entry will appear in a window in the upper right portion of the table. This window consists of 4 lines, each 7 characters in length, and will show the way the description will look above a reference when printed or displayed in expanded mode.

This page is intentionally left blank.

Section 5: Variable Declaration Table

Program annotation can be entered in a program using the variable declaration table, as described in this section.

You may enter nicknames in the program or any subroutine. Nicknames in the program's _MAIN (global) variable declaration table are known to all subroutine blocks. Nicknames in a subroutine block's (local) declaration table are known only to that subroutine.

Each subroutine block may have its own local use of nicknames. The same reference may have different local nicknames in different subroutine blocks, as shown in these examples:

```
BLOCK A     %R1     Light_1
BLOCK B     %R1     Light_2
```

Two subroutine blocks may have the same nickname for different references, as shown in these examples:

```
BLOCK A     %L1     RESET
BLOCK B     %L2     RESET
```

When using a reference address in a program block, Logicmaster 90-30/20/Micro software retrieves the nickname from the local table. If it does not have a local nickname, the software retrieves the nickname from the program's _MAIN table. If the nickname is not in the program's _MAIN table, the software then looks at the reserved nicknames (e.g., FST_SCN, %S0001).

Note

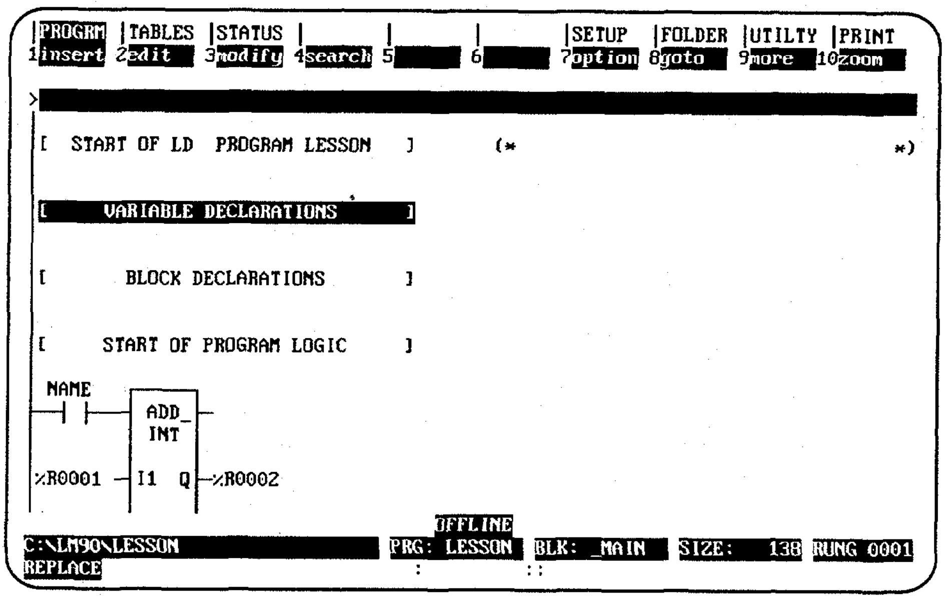
5000 declarations (variables and identifiers) are allowed in the program's _MAIN variable declaration table, and 256 declarations (variables and identifiers) are allowed in each subroutine block's variable declaration table. Each table is always arranged in sorted reference address order.

As more nicknames are used, performance in the program editor may be affected. To enhance performance, 736K of Expanded Memory (LIM/EMS Version 3.02 or later) or SMARTDRV may be used.

Displaying the Variable Declaration Table

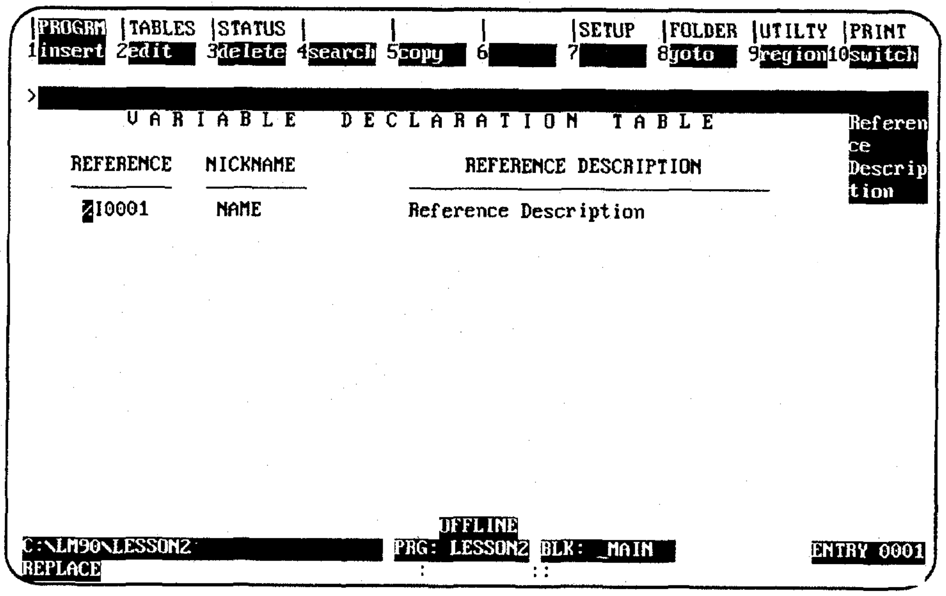
To display the variable declaration table:

1. Move the cursor to highlight the **[VARIABLE DECLARATIONS]** marker.



```
|PROGRM |TABLES |STATUS |        |        |        |SETUP   |FOLDER  |UTILTY  |PRINT
1insert 2edit   3modify 4search 5        6        7option  8goto    9more   10zoom
>
[  START OF LD  PROGRAM LESSON   ]      (*                                        *)

[      VARIABLE DECLARATIONS     ]

[        BLOCK DECLARATIONS      ]

[     START OF PROGRAM LOGIC     ]
 NAME
--| |---ADD_
        INT
%R0001 -I1  Q-%R0002
                                        OFFLINE
C:\LM90\LESSON                          PRG: LESSON  BLK: _MAIN  SIZE:   138 RUNG 0001
REPLACE                                  :      ::
```

2. Press **Zoom** (**F10**).



```
|PROGRM |TABLES |STATUS |        |        |        |SETUP   |FOLDER  |UTILTY  |PRINT
1insert 2edit   3delete 4search 5copy    6        7        8goto    9region10switch
>
    V A R I A B L E   D E C L A R A T I O N   T A B L E                    Referen
                                                                           ce
 REFERENCE   NICKNAME          REFERENCE DESCRIPTION                       Descrip
                                                                           tion
 %I0001      NAME          Reference Description

                                        OFFLINE
C:\LM90\LESSON2                         PRG: LESSON2 BLK: _MAIN           ENTRY 0001
REPLACE                                  :      ::
```

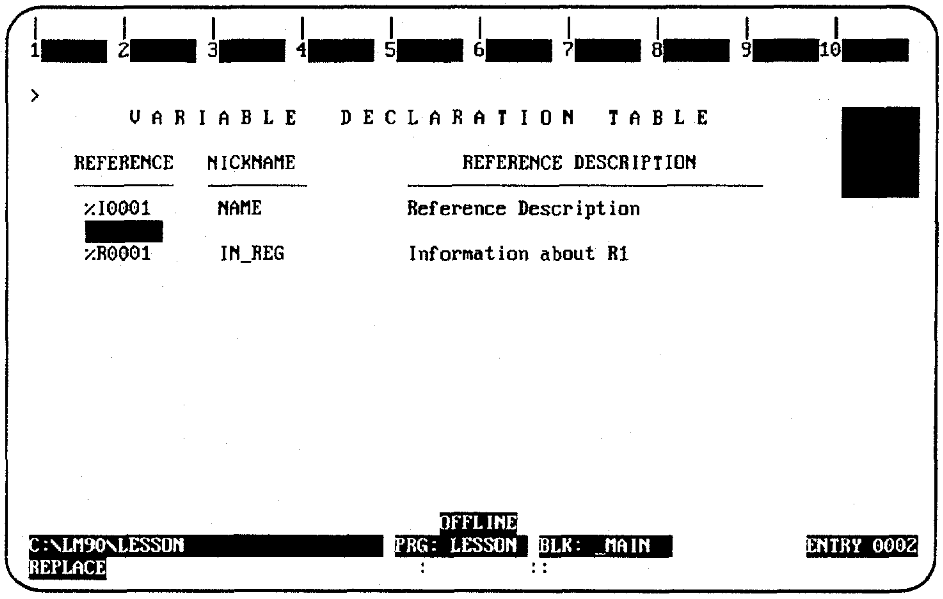
3. The identifier table can be displayed from this screen by pressing **Switch** (**F10**). This table lists the program name, JUMPs, LABELs, MCRs, ENDMCRs, and subroutine blocks declared in this folder. You cannot insert new entries on this display screen; however, you can edit entries already displayed. For example, you could assign an identifier description to the program name.
4. Variable declarations may be viewed in a window on the screen, without leaving the insert or edit function, by pressing **ALT-V** (**ALT-V** once for local variables, **ALT-V** twice for global variables). The variable declarations displayed in this window, however, cannot be edited. Entries temporarily listed (those created or modified during the current editing session) in the variable declarations table can be easily identified by an asterisk (*). Use the Up/Down cursor keys or Page Up/Down keys to scroll through the entries in this window; then, press the **Escape** key to exit the window.

Entering Variable Declarations

To enter new nicknames and reference descriptions as you go:

1. Press **Insert** (**F1**). A field will appear in the reference column.
2. Enter the reference in this field (e.g., %R2 or 2R), and then press the **Enter** or **Tab** key to move to the *Nickname* field.
3. Enter a nickname (e.g., IN_REG), and then press the **Enter** or **Tab** key to move to the *Reference Description* field.
4. Enter a 1- to 32-character description of this reference. The Logicmaster software breaks reference descriptions on boundaries after every seventh character for display in a 4-line x 7-character window. Only 28 characters of the 32-character reference description are displayed in the window. You may want to add extra space in the reference description in order to have the words correctly separated in the window.

   To accept this entry, press the **Enter** key, or the **Plus** (+) key on the numeric keypad. The variable declaration table is automatically sorted each time you press the **Enter** key to accept a new entry. The cursor then moves to the *Reference* field on the next line.



```
 1      2      3      4      5      6      7      8      9      10
>
        V A R I A B L E   D E C L A R A T I O N   T A B L E

     REFERENCE   NICKNAME          REFERENCE DESCRIPTION

      %I0001      NAME             Reference Description

      %R0001      IN_REG           Information about R1



                                            OFFLINE
C:\LM90\LESSON                      PRG: LESSON  BLK: _MAIN          ENTRY 0002
REPLACE                               :            ::
```

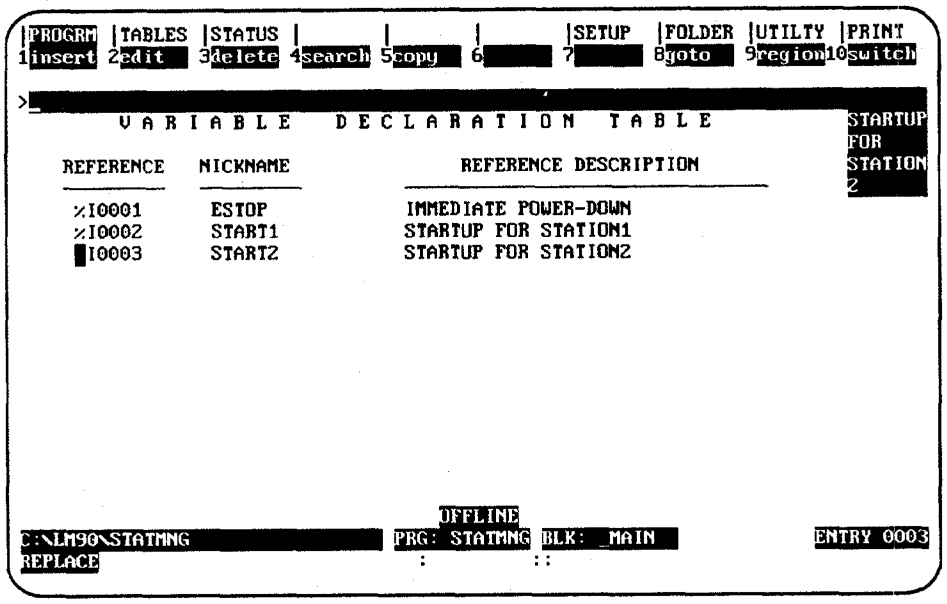
5. To enter a nickname for the next reference, simply press the **Enter** or **Tab** key and the next reference in sequence (in this example, %R0002) will be inserted into the *Reference* field. The cursor will move to the *Nickname* field. To enter a nickname for a different reference, simply type in this reference and press the **Enter** key.
6. You can continue this process until all references have been defined.
7. To exit **INSERT** mode, press the **Escape** key. To exit the variable declaration table and return to the logic entry screen, press **Escape** again.

Copying a Variable Declaration

You can copy text from the NICKNAME and REFERENCE DESCRIPTION of another variable. This editing feature is particularly useful when you are creating a program that has several similar variables using similar nicknames and reference descriptions as in the example shown on this page.

To use this feature, follow these steps:

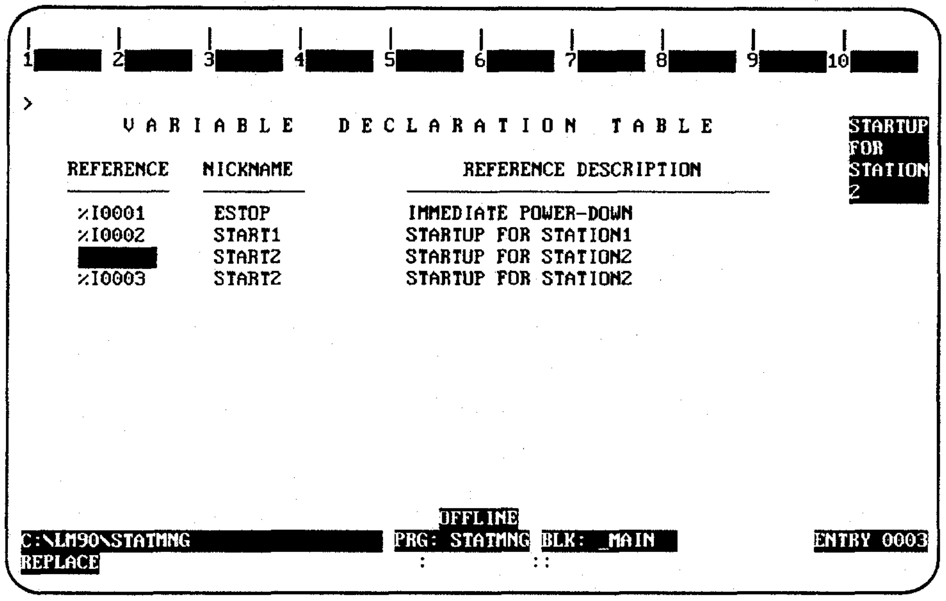
- Enter the line you wish to copy in the standard way.
- After you press **Enter** at the end of the reference description, press the **Escape** key to accept that variable and change the selections available through the function keys.
- Press the **Up Arrow** key (i.e., the upward cursor mover key) once to move the cursor back to the line you just entered.
- Press the **Copy** function key (**F5** as can be seen in the sample shown below).



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3delete 4search 5copy   6       7       8goto   9region10switch
>
        V A R I A B L E    D E C L A R A T I O N   T A B L E          STARTUP
                                                                      FOR
   REFERENCE   NICKNAME            REFERENCE DESCRIPTION              STATION
                                                                      2
    %I0001      ESTOP          IMMEDIATE POWER-DOWN
    %I0002      START1         STARTUP FOR STATION1
    %I0003      START2         STARTUP FOR STATION2

                                          OFFLINE
C:\LM90\STATMNG                         PRG: STATMNG BLK: _MAIN        ENTRY 0003
REPLACE                                   :          ::
```

The line you copied moves down one, and a copy of it appears where your cursor is located (awaiting your modifications). First enter a new reference number, then edit the nickname and description.

The sample shown on the next page shows the screen that appears immediately after pressing the **Copy** key (**F5**) from the screen shown above.



```
1      2      3      4      5      6      7      8      9      10
>
    V A R I A B L E   D E C L A R A T I O N   T A B L E              STARTUP
                                                                      FOR
    REFERENCE   NICKNAME             REFERENCE DESCRIPTION            STATION
                                                                      2
     %I0001      ESTOP          IMMEDIATE POWER-DOWN
     %I0002      START1         STARTUP FOR STATION1
                 START2         STARTUP FOR STATION2
     %I0003      START2         STARTUP FOR STATION2

                                        OFFLINE
C:\LM90\STATMNG                       PRG: STATMNG BLK: _MAIN        ENTRY 0003
REPLACE                                   :         : :
```

Notice that the cursor is resting in the REFERENCE (i.e., reference number) field so that you can assign a unique reference number to it. You will also need to assign a unique NICKNAME and REFERENCE DESCRIPTION. Remember, you can use **`Ctrl-Right Arrow`**, i.e., hold the **Control** key down and press the **Right Arrow** (or Right cursor key), to move the cursor across the letters or words you want to keep. Then key over or add to the text you want to change; e.g., in the example shown above, you would only need to key over the last digit on both the nickname and the description.

Note

Make sure you enter a unique reference number. If you just press the **`Right Arrow`** key to advance to the NICKNAME field, the reference number from the line you copied will drop in by default which will force you to change it to an unused number when you press **`Enter`** at the end of that line.

Editing Variable Declarations

To change the content of a variable declaration:

1. Place the cursor at the declaration to be changed, and press **`Edit`** (**`F2`**).
2. Use the **`Enter`** key, cursor keys, or **`Tab`** key to move from field to field. Type over the entry as needed.
3. After changing the entry, press the **`Enter`** key or the **`Plus`** (**+**) key on the numeric keypad to accept the changes and move to the next table entry to continue editing. Press the **`Escape`** key to accept the changes and terminate the editing session.

The region functions, described in the rung edit section of this chapter, can also be used to select, cut, paste, include, write, and delete variable declarations. For more information on these functions, refer to chapter 3, section 9, "Rung Edit."

Deleting Variable Declarations

To remove one or more entries from the variable declaration table, place the cursor at the first declaration to be deleted and press **`Delete`** (**`F3`**) or **`ALT-D`**. Another way to delete entries from the table is to use the **`Select`** (**`F1`**) softkey to select the variable declaration you wish to delete, then press the **`Delete`** (**`F6`**) edit softkey. Repeat this procedure until all the entries you wish to delete have been removed from the table.

To undo the delete, press **`ALT-A`** before leaving the table. Once you leave a table, you *cannot* undo a deletion made within that table.

Searching for Variable Declarations

The variable declaration table may be searched for a reference or nickname; it cannot be searched for a reference description. To initiate a search in the variable declaration table:

1. Press **`Search`** (**`F4`**) to display the search function window.
2. Enter either the reference or its associated nickname into the *Search for* field. You cannot enter a reference description in this field. Then, press the **`Enter`** key.

Using Goto

The Goto (F8) function key may be used to move the cursor within the variable declaration table. To move to the nth variable declaration, enter **`n`** on the command line, and press Goto (F8). For example, to move the cursor to the first variable declaration, enter **`1`** on the command line, and press **`Goto`** (**`F8`**).

Cut/Pasting Variable Declarations

The region functions, described in the rung edit section of this chapter, can also be used to select, cut, paste, include, write, and delete variable declarations. For more information on these functions, refer to chapter 3, section 9, "Rung Edit."

Note

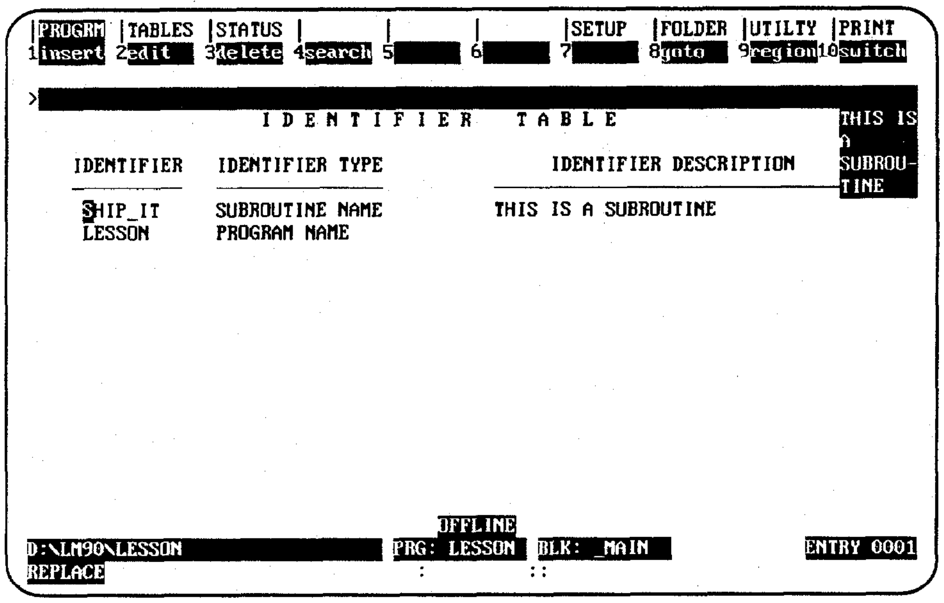
You may not paste or include variable declarations if a reference address in the paste buffer or include file is already in the variable declaration table. However, you may edit the variable declaration table before the paste or include operation to eliminate any reference address conflicts.

Automatically Inserting References

References which do not have nicknames or reference descriptions can be automatically inserted into the variable declaration table as the program is being developed. Then, at a later time, you can go to the variable declaration editor and enter just the annotation. For information on automatically inserting references, refer to chapter 3, section 10, "Editor Options."

Viewing the Identifier Table

The identifier table may be displayed by pressing **Switch** (**F10**) from the variable declaration table.



```
>
                    I D E N T I F I E R   T A B L E           THIS IS
                                                              A
IDENTIFIER   IDENTIFIER TYPE           IDENTIFIER DESCRIPTION SUBROU-
                                                              TINE
SHIP_IT      SUBROUTINE NAME      THIS IS A SUBROUTINE
LESSON       PROGRAM NAME

                                  OFFLINE
D:\LM90\LESSON                    PRG: LESSON  BLK: _MAIN          ENTRY 0001
REPLACE                               :            : :
```

The identifier table in the _MAIN program contains the program name, subroutine block names, JUMPs, LABELs, MCRs, and ENDMCRs. The identifier table in a subroutine block contains JUMPs, LABELs, MCRs, and ENDMCRs.

You cannot insert new entries on this display screen. The program name cannot have its identifier edited. A subroutine block can have its identifier edited if it is not called. All other identifiers and descriptions can be edited.

The identifier table may be searched for a reference or nickname; it cannot be searched for a reference description. To initiate a search in the identifier table:

1. Press **Search** (**F4**) to display the search function window.
2. Enter either the reference or its associated nickname into the *Search for* field. You cannot enter a reference description in this field. Then, press the **Enter** key.

Importing to and Exporting from the Variable Declarations Table

Comma Separated Variable (CSV) Format

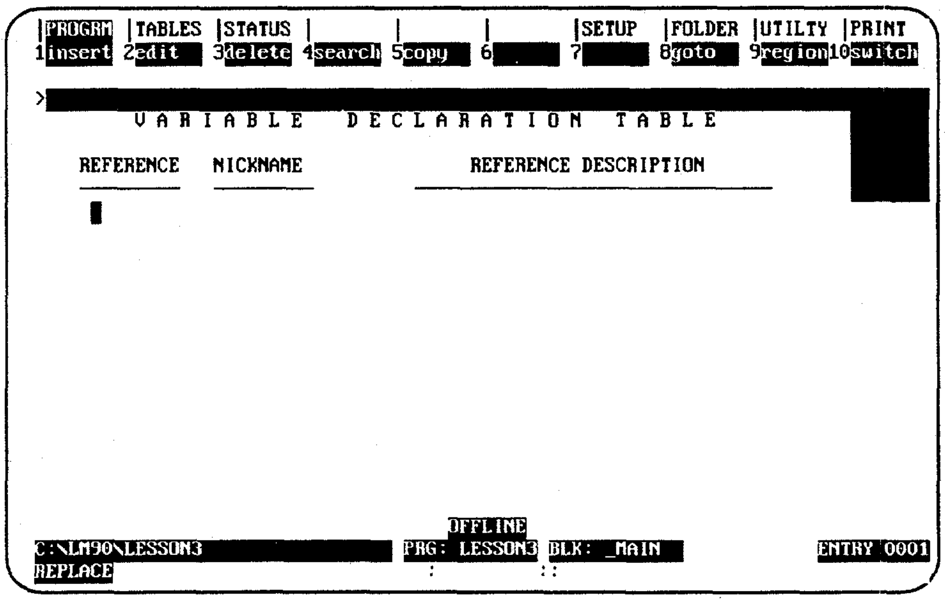
For Logicmaster uses an extension of the industry-standard Comma Separated Variable (CSV) format called Shared Name File (SNF) format. Importing an SNF into the Variable Declarations Table gives you the ability to define nicknames ahead of time in a spreadsheet program. Exporting an SNF from the Variable Declarations Table gives you the ability to use the exported file with CIMPLICITY® and third party operator interfaces.

The steps for importing and exporting files that use the SNF format are shown here. For information about SNF format, see Appendix I.

Importing SNF Formatted Files

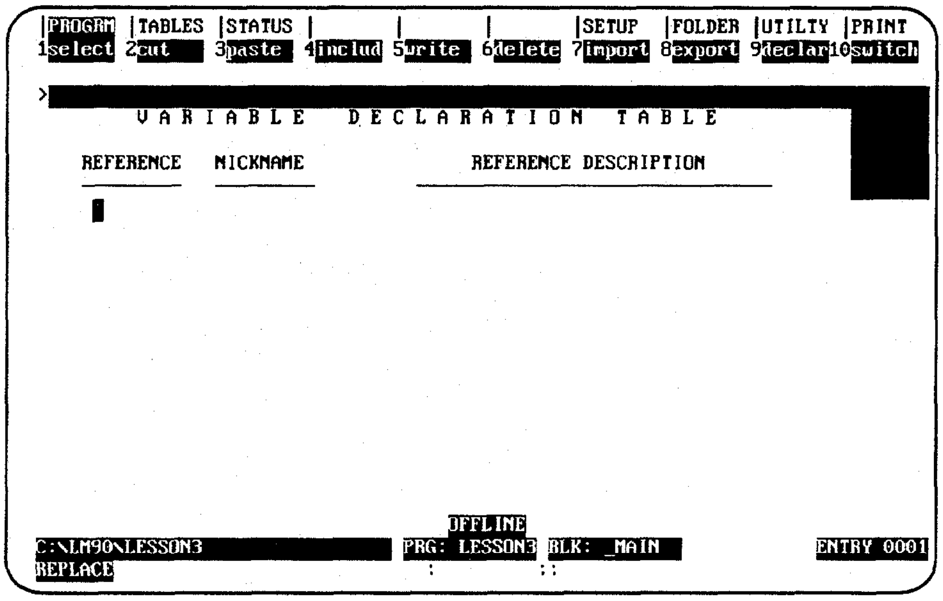
When importing Shared Name File (SNF) format files, the SNF files must adhere to the standards discussed in Appendix I of this manual. If your file does adhere to those standards, then follow these steps to import:

1. Before you begin the next steps (which are all from within the programming software), make sure that your SNF is in the same directory as the program into which you are going to import.
2. Move the cursor to highlight the **[VARIABLE DECLARATIONS]** marker as shown on page 3-34.
3. Press **Zoom** (**F10**).



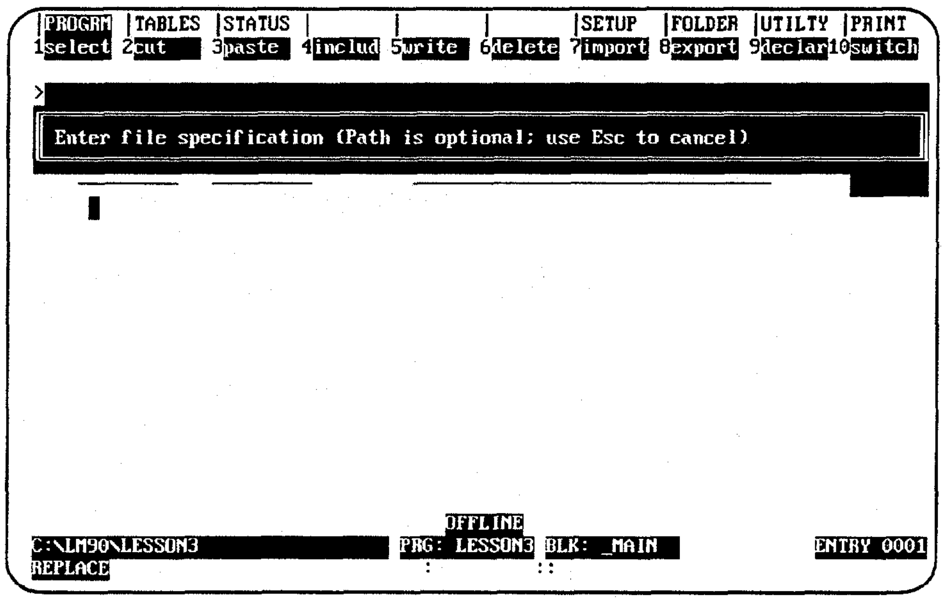
```
|PROGRM |TABLES |STATUS |        |        |        |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3delete 4search 5copy   6       7       8goto   9region10switch
>
         V A R I A B L E   D E C L A R A T I O N   T A B L E

   REFERENCE    NICKNAME              REFERENCE DESCRIPTION
   ---------    --------          ---------------------------



                                      OFFLINE
C:\LM90\LESSON3                       PRG: LESSON3 BLK: _MAIN           ENTRY 0001
REPLACE                                    :          : :
```

4. Press **region** (**F9**). Your screen display will change to the one shown on the following page.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1select 2cut    3paste  4includ 5write  6delete 7import 8export 9declar10switch
>
        V A R I A B L E   D E C L A R A T I O N   T A B L E

     REFERENCE   NICKNAME              REFERENCE DESCRIPTION


                                  OFFLINE
C:\LM90\LESSON3                   PRG: LESSON3 BLK: _MAIN          ENTRY 0001
REPLACE                                 :        ::
```

5. Press **import** (**F7**). You will see the message shown in the following sample screen.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1select 2cut    3paste  4includ 5write  6delete 7import 8export 9declar10switch
>
  Enter file specification (Path is optional; use Esc to cancel)


                                  OFFLINE
C:\LM90\LESSON3                   PRG: LESSON3 BLK: _MAIN          ENTRY 0001
REPLACE                                 :        ::
```

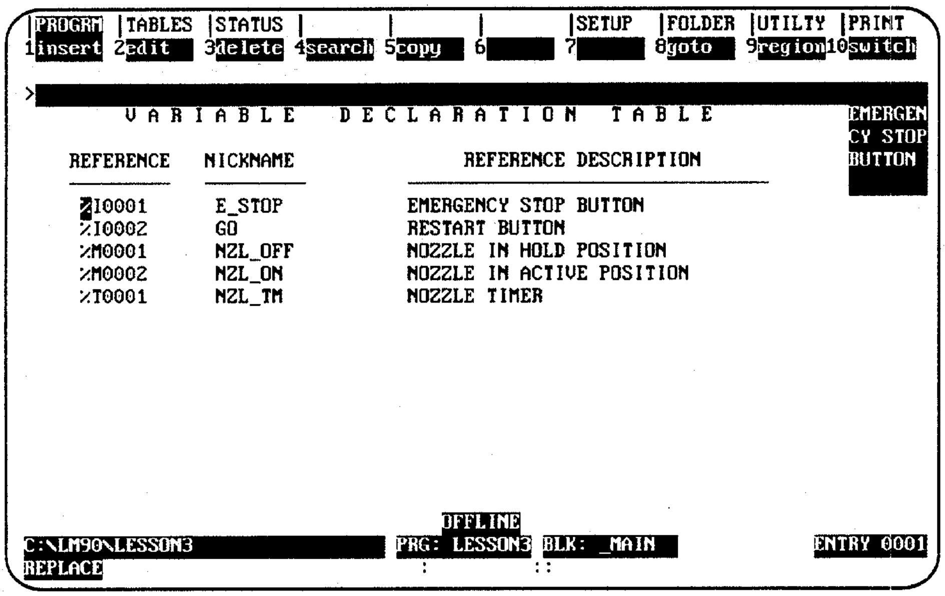
6. Type the file name. The path is optional because in Step 1 you ensured that the SNF was in the same directory as the folder.

   As long as your SNF follows the guidelines discussed in Appendix I, it will import correctly into the Variable Declarations Table.

Exporting SNF (CSV) Formatted Files

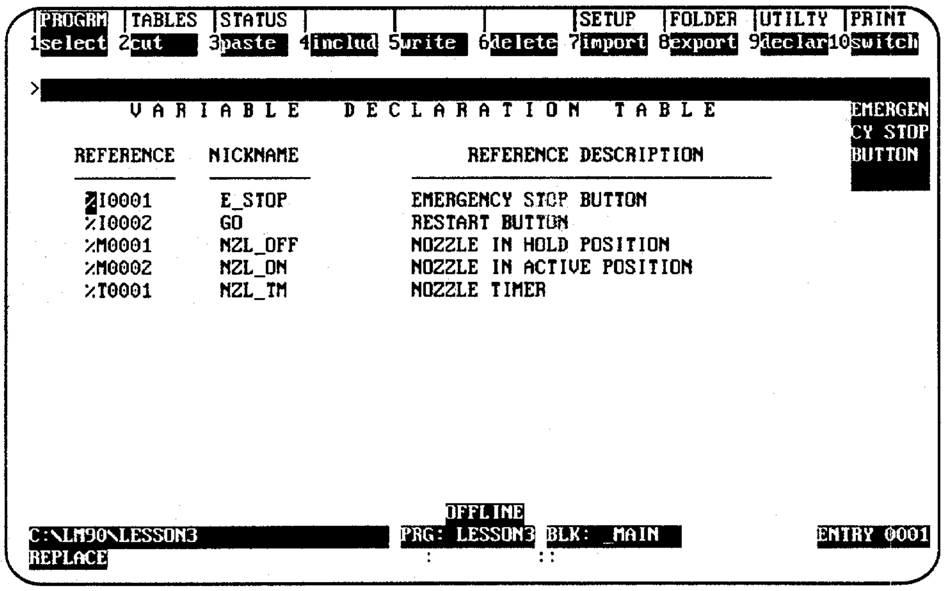
SNF is an extension of the industry-standard Comma Separated Variable (CSV) format. For more information about CSV and SNF formats, refer to page 3-41. When exporting your Variable Declarations Table to SNF format, Logicmaster will put the Variable Declarations into Shared Name File (SNF) format. For information about SNF format, refer to Appendix I of this manual.

1. Move the cursor to highlight the **[VARIABLE DECLARATIONS]** marker as shown on page 3-34.
2. Press **Zoom** (**F10**).



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3delete 4search 5copy   6       7       8goto   9region10switch

>
        V A R I A B L E   D E C L A R A T I O N   T A B L E            EMERGEN
                                                                       CY STOP
   REFERENCE   NICKNAME         REFERENCE DESCRIPTION                  BUTTON

   %I0001      E_STOP           EMERGENCY STOP BUTTON
   %I0002      GO               RESTART BUTTON
   %M0001      NZL_OFF          NOZZLE IN HOLD POSITION
   %M0002      NZL_ON           NOZZLE IN ACTIVE POSITION
   %T0001      NZL_TM           NOZZLE TIMER

                                          OFFLINE
C:\LM90\LESSON3                          PRG: LESSON3 BLK: _MAIN          ENTRY 0001
REPLACE                                     :          : :
```

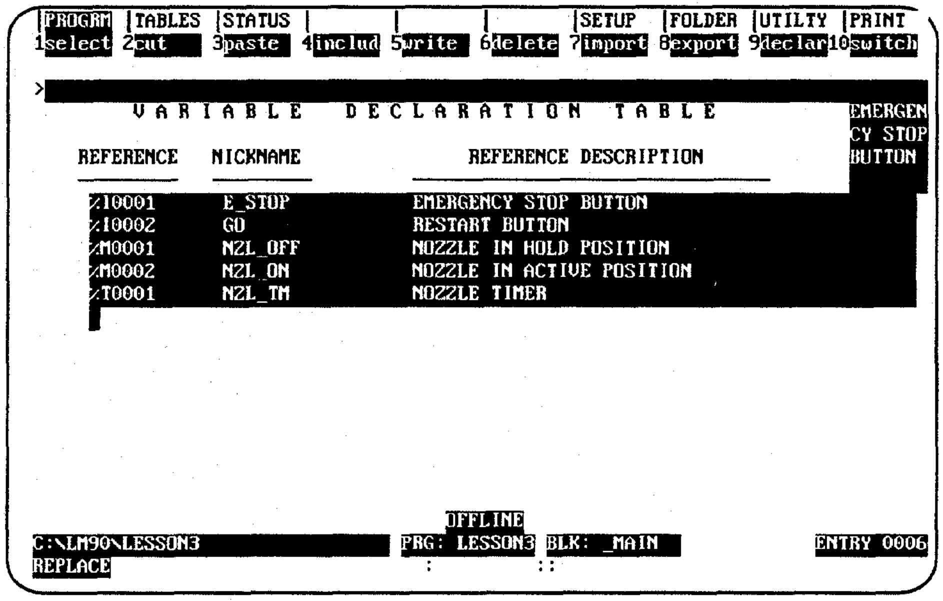
3. Press **region** (**F9**). The selections at the top of the screen will change to the ones shown below.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1select 2cut    3paste  4includ 5write  6delete 7import 8export 9declar10switch

>
        V A R I A B L E   D E C L A R A T I O N   T A B L E            EMERGEN
                                                                       CY STOP
   REFERENCE   NICKNAME         REFERENCE DESCRIPTION                  BUTTON

   %I0001      E_STOP           EMERGENCY STOP BUTTON
   %I0002      GO               RESTART BUTTON
   %M0001      NZL_OFF          NOZZLE IN HOLD POSITION
   %M0002      NZL_ON           NOZZLE IN ACTIVE POSITION
   %T0001      NZL_TM           NOZZLE TIMER

                                          OFFLINE
C:\LM90\LESSON3                          PRG: LESSON3 BLK: _MAIN          ENTRY 0001
REPLACE                                     :          : :
```

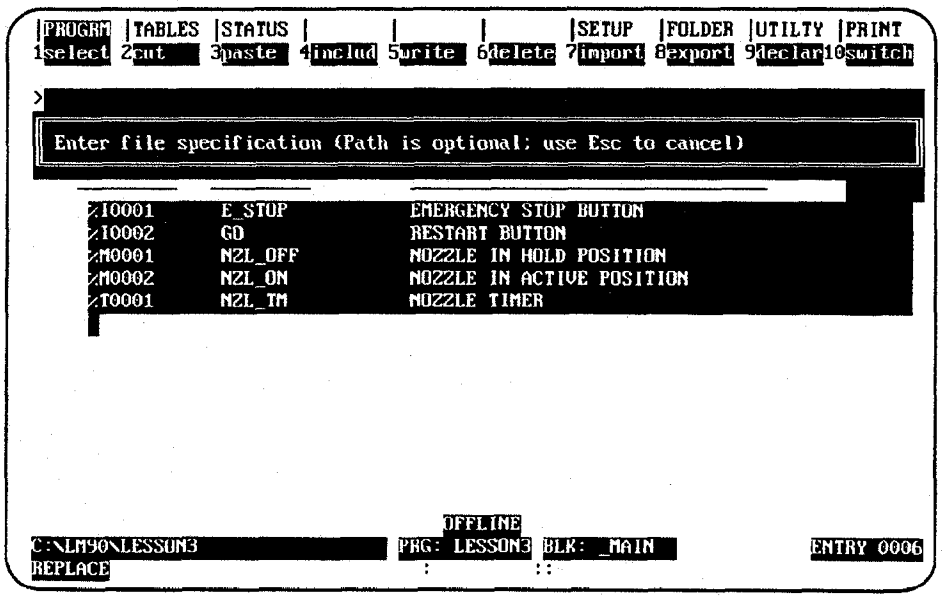
4. Press **select** (**F1**). Then press the **Cursor Down** key (the **Down Arrow** key) until you have highlighted the portion of the Variable Declaration Table you wish to export.

The selected region will appear in reverse video as represented in the sample shown below.


|PROGRM |TABLES |STATUS | | | |SETUP |FOLDER |UTILTY |PRINT
1select 2cut 3paste 4includ 5write 6delete 7import 8export 9declar10switch

```
>
      V A R I A B L E   D E C L A R A T I O N   T A B L E          EMERGEN
                                                                    CY STOP
   REFERENCE   NICKNAME            REFERENCE DESCRIPTION            BUTTON

   %I0001      E_STOP         EMERGENCY STOP BUTTON
   %I0002      GO             RESTART BUTTON
   %M0001      NZL_OFF        NOZZLE IN HOLD POSITION
   %M0002      NZL_ON         NOZZLE IN ACTIVE POSITION
   %T0001      NZL_TM         NOZZLE TIMER

                                   OFFLINE
C:\LM90\LESSON3               PRG: LESSON3 BLK: _MAIN           ENTRY 0006
REPLACE                           :           : :
```

5. Press **export** (**F8**); then type the name you wish to call the SNF that Logicmaster will create from your Variable Declarations. You do not have to add an extension; Logicmaster will automatically add the extension .SNF. (The recommended method of nomenclature is to use the same name for your SNF as the folder from which it came.) If you do not enter a path, the file will save in the same directory where your program resides. Refer to Appendix I for more information about SNF format.


|PROGRM |TABLES |STATUS | | | |SETUP |FOLDER |UTILTY |PRINT
1select 2cut 3paste 4includ 5write 6delete 7import 8export 9declar10switch

```
>
 Enter file specification (Path is optional; use Esc to cancel)

   %I0001      E_STOP         EMERGENCY STOP BUTTON
   %I0002      GO             RESTART BUTTON
   %M0001      NZL_OFF        NOZZLE IN HOLD POSITION
   %M0002      NZL_ON         NOZZLE IN ACTIVE POSITION
   %T0001      NZL_TM         NOZZLE TIMER

                                   OFFLINE
C:\LM90\LESSON3               PRG: LESSON3 BLK: _MAIN           ENTRY 0006
REPLACE                           :           : :
```

6. After typing in a name, press **Enter**. The words "Export completed" will appear in the upper left portion of your screen just below the menu.
7. Press the **Escape** key after seeing the "Export completed" to return to the program.

This page is intentionally left blank.

Section 6: Rung Comments

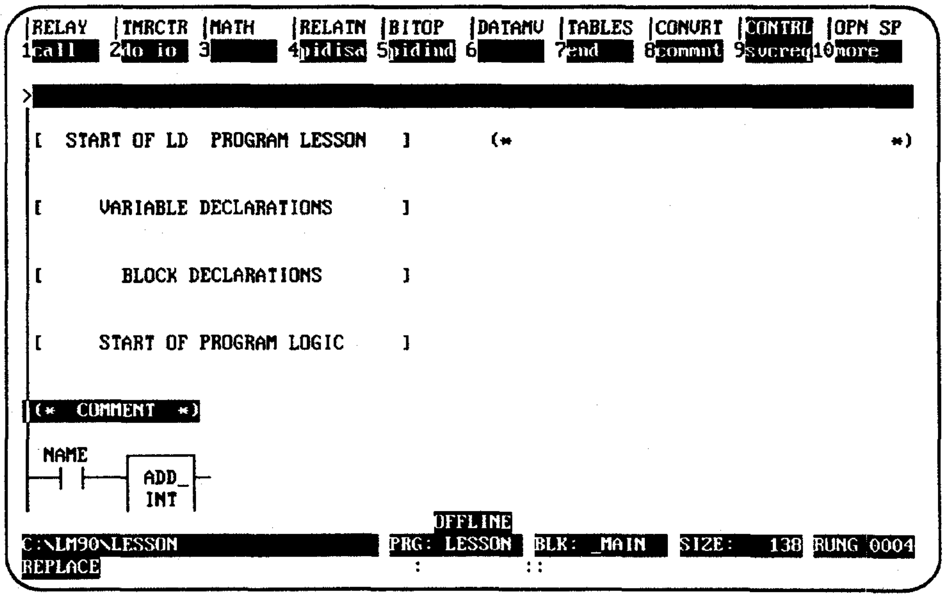
Each COMMENT instruction has a unique text block associated with it. Pasting/including a comment will create a copy of the text block for each COMMENT instruction pasted. The text for any COMMENT instruction is edited independently of any other COMMENT instruction. This allows you to copy a COMMENT instruction by cutting and pasting the text; then, you can edit the pasted text. Refer to chapter 3, section 5, "Variable Declaration Table," for information on cutting and pasting variable declarations.

Release 1 and Release 2 of Logicmaster 90-30/20/Micro software permitted two or more comments to be associated with one block of text. If you have such duplicate comments and want to separate them, write the entire block contents to a side file and delete all the logic. Then, include the side file into the empty block. Now, each of the duplicate comment locations will have its own copy of the text.

Inserting a Rung Comment

To insert rung comments, move the cursor to the rung you wish to insert a comment before.

1. Press **Insert** (**F1**) and then **Control** (**Shift-F9**) to select the control functions.
2. Press **F8** to select the COMMENT function, or use the mnemonic by typing: **&COMMENT** on the command line and pressing the **Enter** key. The screen should appear as shown below:

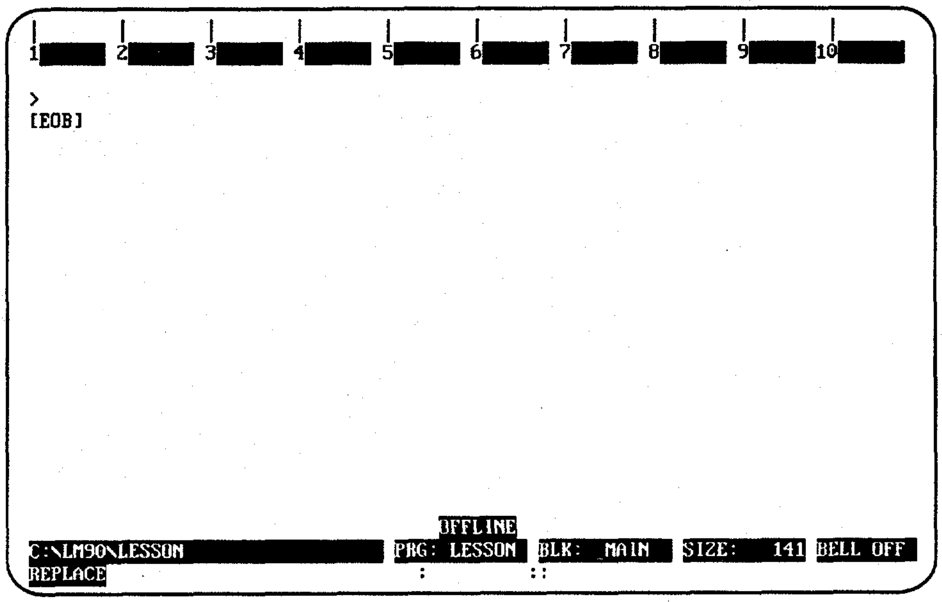


```
|RELAY  |TMRCTR |MATH    |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1call   2do io  3        4pidisa 5pidind 6       7end    8commnt 9svcreq10more

>

[  START OF LD  PROGRAM LESSON    ]        (*                                   *)

[     VARIABLE DECLARATIONS       ]

[       BLOCK DECLARATIONS        ]

[     START OF PROGRAM LOGIC      ]

(*  COMMENT  *)
 NAME
-| |---[ADD_ INT]-
                                         OFFLINE
C:\LM90\LESSON                          PRG: LESSON  BLK: _MAIN  SIZE:    138 RUNG 0004
REPLACE                                     :          ::
```

3. Press the **Escape** key to accept the rung and exit **INSERT** mode.

Adding Text

Text can be entered into the COMMENT instruction by positioning the cursor on the COMMENT instruction and pressing **Zoom** (**F10**).



```
|    |    |    |    |    |    |    |    |    |
1    2    3    4    5    6    7    8    9    10

>
[EOB]

                                  OFFLINE
C:\LM90\LESSON                    PRG: LESSON  BLK: MAIN  SIZE:   141  BELL OFF
REPLACE                                :          ::
```

This is a simple full-screen editor, which allows you to input your rung description. Up to 2048 characters of text are permitted.

The available keys in the comment editor include:

Key	Description
Cursor keys	Move the cursor within the rung.
Page Up key	Move the cursor up one page.
Page Down key	Move the cursor down one page.
Insert key	Change the text editing mode (**INSERT** or **REPLACE**).
Delete key	Delete the character at the cursor position.
Backspace key	Delete the character to the left of the cursor position.
Home key	Position the cursor on the first character of the first line of the comment text.
End key	Position the cursor at the end of the comment text.

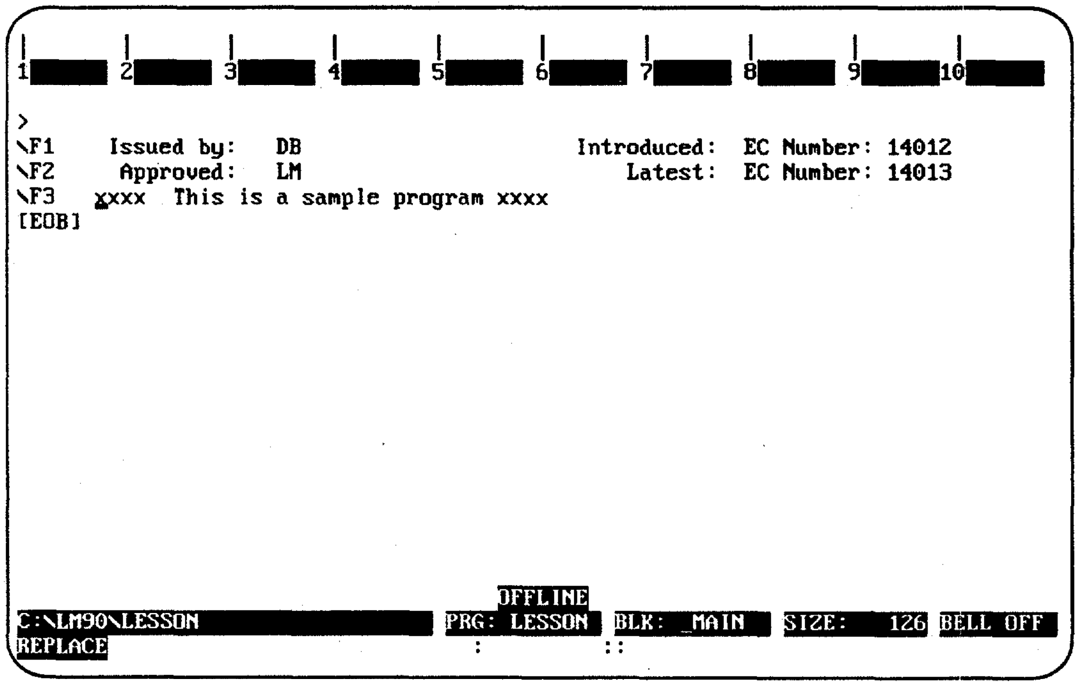
User-Defined Footers in Listings

You can define up to four (4) lines of text to be printed at the bottom of each listing page generated by the Logicmaster 90-30 Programmer printing utility. Specify these lines of text using comment directives similar to the existing title, subtitle and border directives (\T, \S, and \B respectively).

To use create a footer for a program file, follow these steps:

1. Follow the steps described on the previous pages to create a "COMMENT."
2. On a separate line at the beginning of the comment text, key in **\F1** (for Footer 1); then key in the text you wish to appear in the footer. (You can enter up to four footers in this manner, using F1, F2, F3, and F4 as the footer line directives for footers 1 through 4 respectively. You can enter them in any order, but each must be on a separate line and must be at the beginning of the comment text in which they are entered. Regardless of the order entered, they will always print with Footer 1 before Footer 2, Footer 2 before Footer 3, etc.)

   A sample of this is shown below:



```
>
\F1    Issued by:   DB                         Introduced:  EC Number: 14012
\F2     Approved:   LM                             Latest:  EC Number: 14013
\F3   xxxx  This is a sample program xxxx
[EOB]
```

3. When you are finished with your footers, press the **Escape** key to return to your program and accept that rung.

Note

Footer 4 supercedes the standard LM90 listing page footer line. If \F4 is not specified, the standard Logicmaster™ 90 footer is printed. If the \F4 directive is specified without text, the LM90 footer line will not be printed.

The following text shows the footer that resulted from the comment/footer settings shown on the previous page:

```
                Issued by:  DB                    Introduced:  EC Number: 14012
                 Approved:  LM                        Latest:  EC Number: 14013
                         xxxx  This is a sample program  xxxx
Program: STATMNG                    C:\LM90\STATMNG                    Block: _MAIN
```

The following restrictions should be observed:

- Any blank footer lines will be printed in the listing as blanks.
- If text for a specified footer line has been previously defined and that footer line directive is specified again, but without text, that footer line will no longer be printed.
- Once specified, footer lines will be printed on each page of the listing until they are changed with user-defined footer line directives in subsequent comment text.
- For an 80-column listing, you can enter text up to 80 characters in length. For a 132-column listing, you can enter up to 132 characters.
- Footer line text less than the page width is centered to the page.
- To start user-defined footers on the pages prior to the logic, the comment containing the user-defined footers must be the first instruction in the main block of the program.

Creating Borders

To make comments stand out in program printouts, borders can be printed around them. The software will automatically create the border using any character you enter, as instructed below:

1. Create the text for the comment.
2. At the beginning of the comment on a line by itself, enter **\B** (or **\b**, as lowercase alphanumeric characters are also recognized) and an ASCII character to be used as the border. The ASCII character must be the next character after the B. For example, entering **\B?** would create a border of question marks around the outside of the text. If the comment text does not contain **\B** followed by the ASCII character for the border, then Logicmaster 90-30/20/Micro software prints asterisks by default.

```
 12/2/93  11:49        GE FANUC LOGICMASTER 90-30 DOCUMENTATION             Page 3
                                 TITLE APPEARS HERE
                               SUBTITLE APPEARS HERE
| << RUNG 4   STEP #0049 >>
|
| (???????????????????????????????????????????????????????????????????????????????)
| (? The following logic rung enables automatic mode when all enabling          ?)
| (? conditions are met.                                                        ?)
| (???????????????????????????????????????????????????????????????????????????????)
|
| <<RUNG 5   STEP #0050 >>    Cross reference for AUTO
|                             |/|   5, 7
|
| AUTOPB  LISUP   UNISUP %Q0012  RSTAHPB EMERGST %Q0006  COOLANT                AUTO
+--| |-----| |-----| |-----| |--+--|/|-----|/|-----|/|-----|/|------------------( )
|           9       10      19  |
| AUTO                          |
+---|/|-------------------------+
|    5
|
| << RUNG 6   STEP #0060 >>    Cross reference for EMERGST
|                              | |   5, 6
|
| EMSTOP  START  CLAMPED %I0073                                            EMERGST
+---| |--+--| |--+--| |--+--| |-------------------------------------------------( )
|        |       |  29   |
|        |EMERGST|SPN MTR|
|        +--|/|--+--|/|--+
|            6       8
|
```

The border of question marks shown in this example will be used for all subsequent rung comments in the program, unless it is changed or the border is deleted.

3. To print out a comment with no border, enter **\B** followed by the space character.

Starting a New Page of Comments

To print a new page of comments:

1. Create the text for the comment.
2. Enter **`\P`** or **`\p`**, as lowercase alphanumeric characters are also recognized, on a line by itself.

The text following the **`\P`** character will then begin on a new page.

The **`\P`** can be used several times within a comment, with text before and after each one.

Printing a Title

When the printer output is defined as described in chapter 9, a title and subtitle for the printout are created. This title and subtitle will appear on every page, unless changed as described below:

1. At the place in the program where the new title and subtitle should begin, enter a comment in the logic. This will be printed as the beginning of a new page.
2. On the first line of the comment, enter the following:

   A. **`\T`** or **`\t`** with the new title (up to 62 characters) on the same line.

   B. **`\S`** or **`\s`** with the new subtitle (also up to 62 characters) on the same line.

The new title and subtitle will appear on each subsequent page. **`\T`** or **`\S`** alone can be specified. **`\B`**, **`\T`**, or **`\S`** must be on separate lines. Those lines must be first in a comment, but they can be in any order.

Creating Longer Comments

The maximum number of characters that can be included in a comment is 2048. Longer text can be included in printouts (but not displayed in program function or mode) using an annotation text file, as described below:

1. Create the comment as described on the previous page.

   A. Enter text to the point where the text from the other file should begin.

   B. Move the cursor to the beginning of a new line and enter **`\I`** or **`\i`**, the drive followed by a colon, the subdirectory or folder, and the file name, as shown in this example:

   ```
   \I d:\text\commnt1
   ```

   The drive designation is not necessary if the file is located on the same drive as the program folder.

   C. Continue editing the program, or exit to MS-DOS.

2. After exiting the programmer, create a text file using any MS-DOS compatible software package. Give the file the file name in the comment, and place it on the drive specified in the comment.

Section 7: Changing the Display Mode

The view mode setup function, described in chapter 6, "Programmer Setup," enables you to specify which modes are displayed when you press **ALT-N**. These view modes range from showing only rung references to showing reference names and reference descriptions in an expanded rung form (display all mode). You can also view the maximum amount of program logic on a screen by selecting a compressed rung mode.

Display all mode shows reference description information in four 7-character segments at each occurrence of an identifier or reference address in the program editor. The Variable Declaration Editor displays a 4 x 7 character window with the reference description for the current entry.

Note

The display mode (ALT-N) cannot be changed during **RUNG EDIT** or **RUNG INSERT** mode.

Each time **ALT-N** is pressed, the editor display will move to the next display mode. The five display modes are listed below.

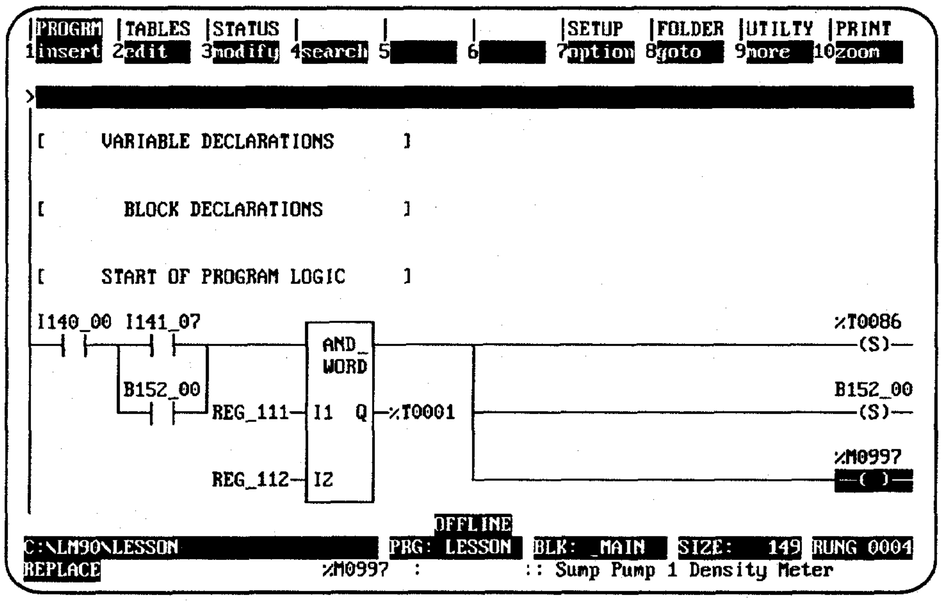
Display All Mode	Description	Lines / Rung
A	*Reference descriptions off, nicknames on.*	*3 screen lines per rung line.*
B	Reference descriptions off, nicknames off.	3 screen lines per rung line.
C	Reference descriptions on, nicknames on.	7 screen lines per rung line.
D	Reference descriptions on, nicknames off.	7 screen lines per rung line.
E	Compressed rung mode.	1 screen line per rung line.

Mode A is the default mode. Each time **ALT-N** is pressed, the mode changes to the next mode (e.g., mode A to B). If **ALT-N** is pressed while mode E is displayed, the display cycles back to mode A. Refer to the information on "View Modes Setup (ALT-N)" in chapter 6, "Programmer Setup," in order to select the modes displayed when **ALT-N** is pressed.

Note

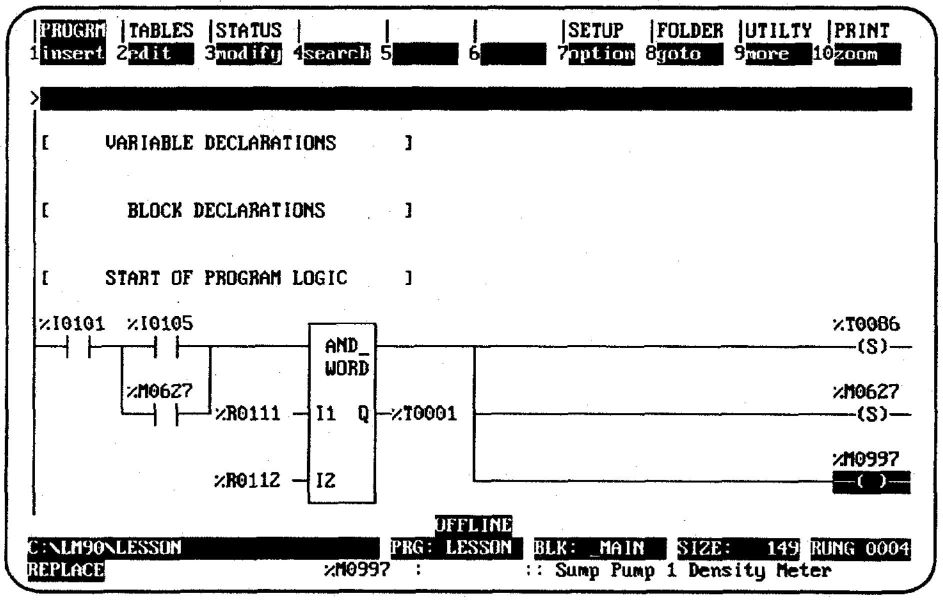
The mode identifier letters A, B, C, D, and E are used for reference in this document only.

Display Mode A: Reference descriptions off, nicknames on.


|PROGRM |TABLES |STATUS | | | |SETUP |FOLDER |UTILTY |PRINT
1insert 2edit 3modify 4search 5 6 7option 8goto 9more 10zoom
AND_
WORD
I1 Q
I2

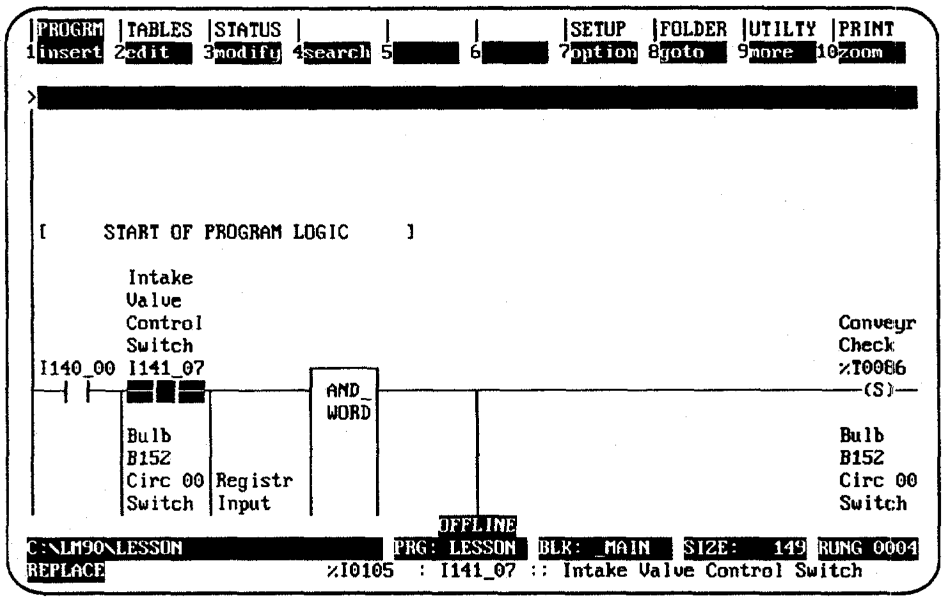
```
[     VARIABLE DECLARATIONS      ]

[       BLOCK DECLARATIONS       ]

[     START OF PROGRAM LOGIC     ]

I140_00 I141_07                                        %T0086
B152_00                                                B152_00
REG_111-I1  Q-%T0001
REG_112-I2                                             %M0997
OFFLINE
C:\LM90\LESSON         PRG: LESSON  BLK: _MAIN  SIZE:   149  RUNG 0004
REPLACE                %M0997   :       :: Sump Pump 1 Density Meter
```

Display Mode B: Reference descriptions off, nicknames off.


|PROGRM |TABLES |STATUS | | | |SETUP |FOLDER |UTILTY |PRINT
1insert 2edit 3modify 4search 5 6 7option 8goto 9more 10zoom
AND_
WORD
I1 Q
I2

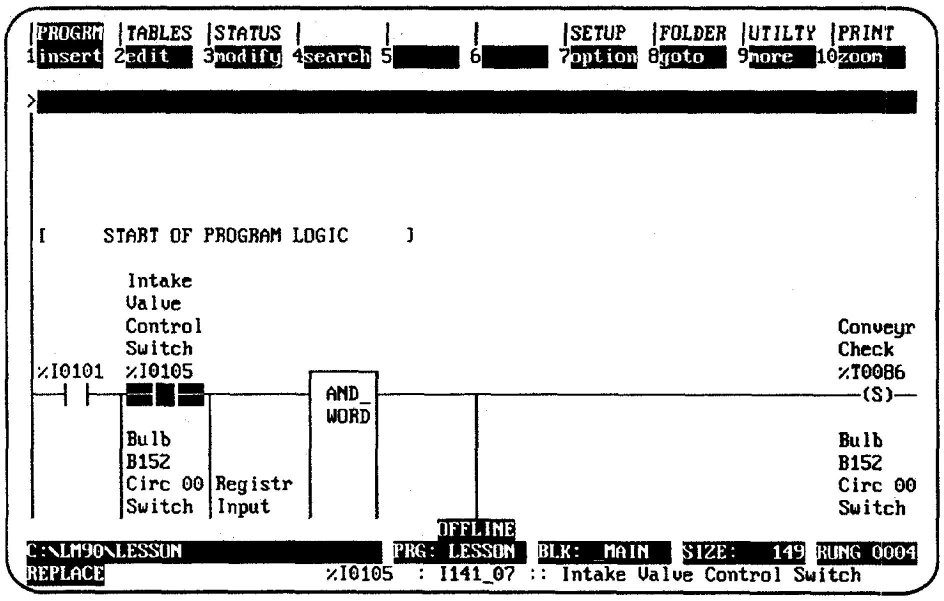
```
[     VARIABLE DECLARATIONS      ]

[       BLOCK DECLARATIONS       ]

[     START OF PROGRAM LOGIC     ]

%I0101  %I0105                                         %T0086
%M0627                                                 %M0627
%R0111 -I1  Q-%T0001
%R0112 -I2                                             %M0997
OFFLINE
C:\LM90\LESSON         PRG: LESSON  BLK: _MAIN  SIZE:   149  RUNG 0004
REPLACE                %M0997   :       :: Sump Pump 1 Density Meter
```

Display Mode C: Reference descriptions on, nicknames on.



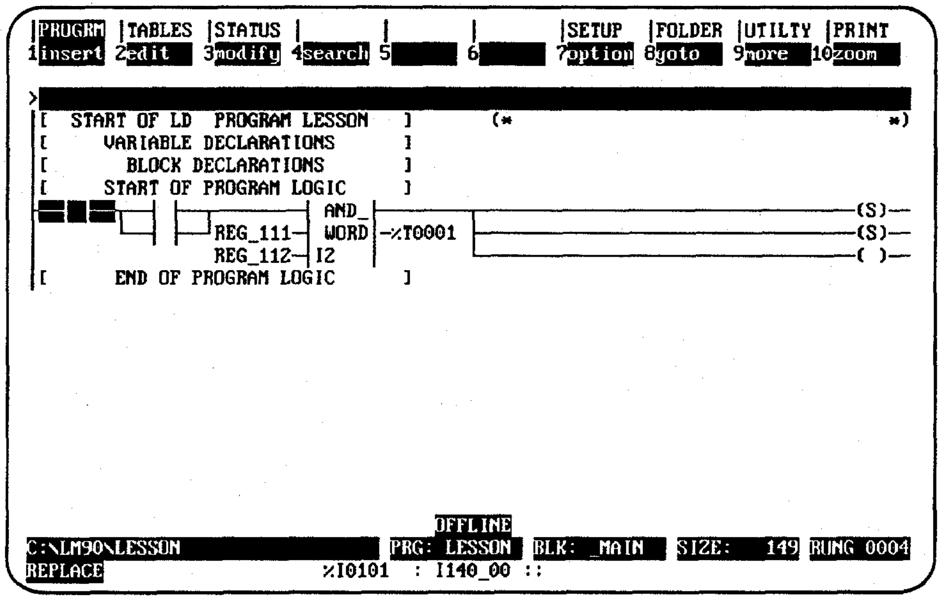
```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3modify 4search 5       6       7option 8goto   9more   10zoom
>

[     START OF PROGRAM LOGIC      ]

         Intake
         Value
         Control                                                  Conveyr
         Switch                                                   Check
I140_00  I141_07                                                  %T0086
--| |---[###]-----------[AND_ ]---------------------------------------(S)--
        |        |       [WORD ]       |
        |Bulb    |       |     |       |                          Bulb
        |B152    |       |     |       |                          B152
        |Circ 00 |Registr|     |       |                          Circ 00
        |Switch  |Input  |     |       |                          Switch
                                    OFFLINE
C:\LM90\LESSON                PRG: LESSON  BLK: _MAIN  SIZE:   149 RUNG 0004
REPLACE                  %I0105   : I141_07 :: Intake Valve Control Switch
```

Display Mode D: Reference descriptions on, nicknames off.



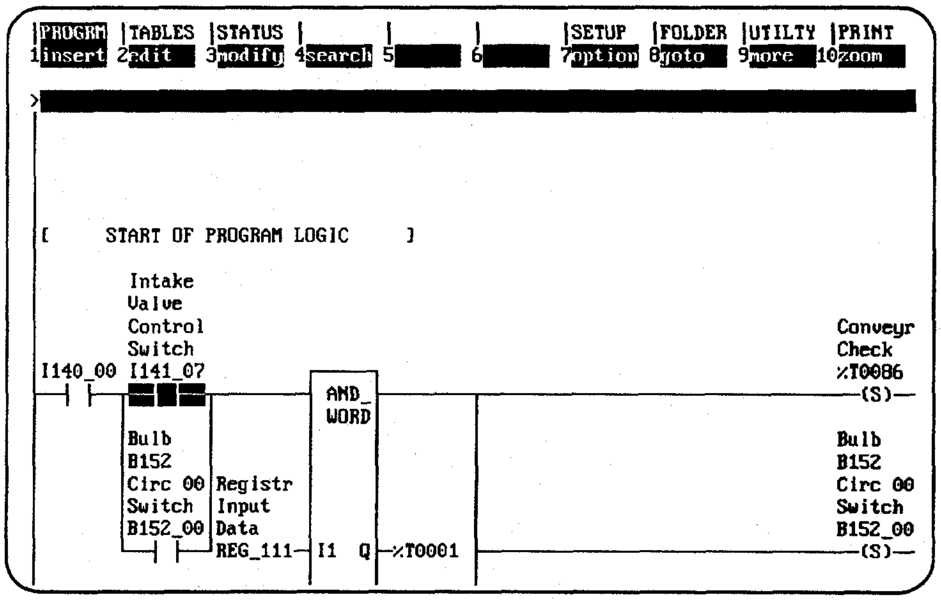
```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3modify 4search 5       6       7option 8goto   9more   10zoom
>

[     START OF PROGRAM LOGIC      ]

         Intake
         Value
         Control                                                  Conveyr
         Switch                                                   Check
%I0101   %I0105                                                   %T0086
--| |---[###]-----------[AND_ ]---------------------------------------(S)--
        |        |       [WORD ]       |
        |Bulb    |       |     |       |                          Bulb
        |B152    |       |     |       |                          B152
        |Circ 00 |Registr|     |       |                          Circ 00
        |Switch  |Input  |     |       |                          Switch
                                    OFFLINE
C:\LM90\LESSON                PRG: LESSON  BLK: _MAIN  SIZE:   149 RUNG 0004
REPLACE                  %I0105   : I141_07 :: Intake Valve Control Switch
```

Display Mode E: Compressed rung mode.



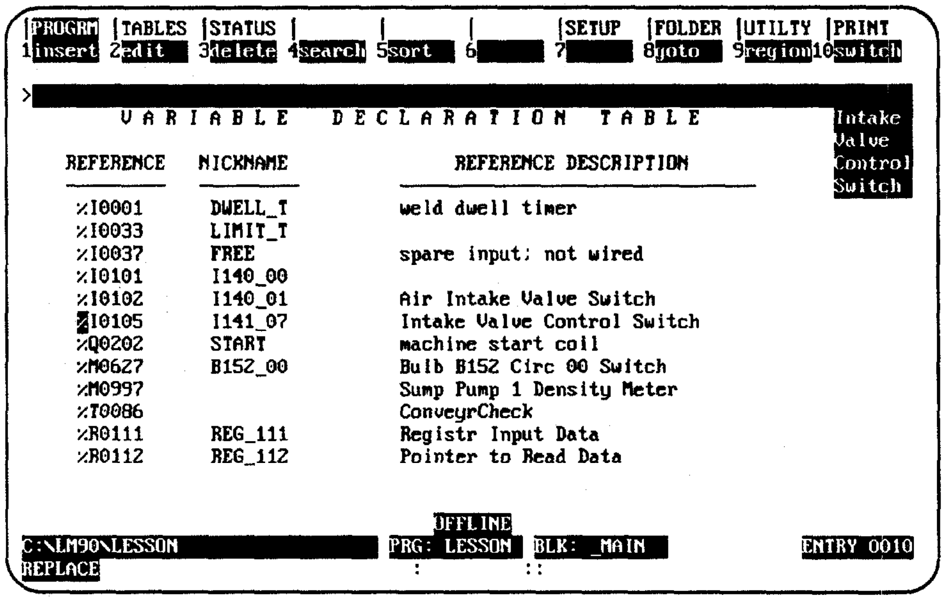
```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3modify 4search 5       6       7option 8goto   9more   10zoom
>
[  START OF LD  PROGRAM LESSON   ]     (*                                    *)
[     VARIABLE DECLARATIONS      ]
[       BLOCK DECLARATIONS       ]
[     START OF PROGRAM LOGIC     ]
                         AND_ 
            REG_111-  WORD -%T0001                                    (S)
            REG_112-  I2                                              (S)
                                                                      ( )
[       END OF PROGRAM LOGIC     ]

                                         OFFLINE
C:\LM90\LESSON                     PRG: LESSON  BLK: _MAIN  SIZE:   149 RUNG 0004
REPLACE                     %I0101  : I140_00 ::
```

Display Mode D without status lines (press **ALT-E** to remove the status lines):



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3modify 4search 5       6       7option 8goto   9more   10zoom
>

[     START OF PROGRAM LOGIC     ]

        Intake
        Valve
        Control                                                  Conveyr
        Switch                                                   Check
I140_00 I141_07                                                  %T0086
                          AND_                                   (S)
                          WORD
        Bulb                                                     Bulb
        B152                                                     B152
        Circ 00 Registr                                          Circ 00
        Switch  Input                                            Switch
        B152_00 Data                                             B152_00
                REG_111-  I1   Q -%T0001                         (S)
```

The following screen shows how the reference description breaks within the 4-line x 7-character window when the cursor is on %I0105.



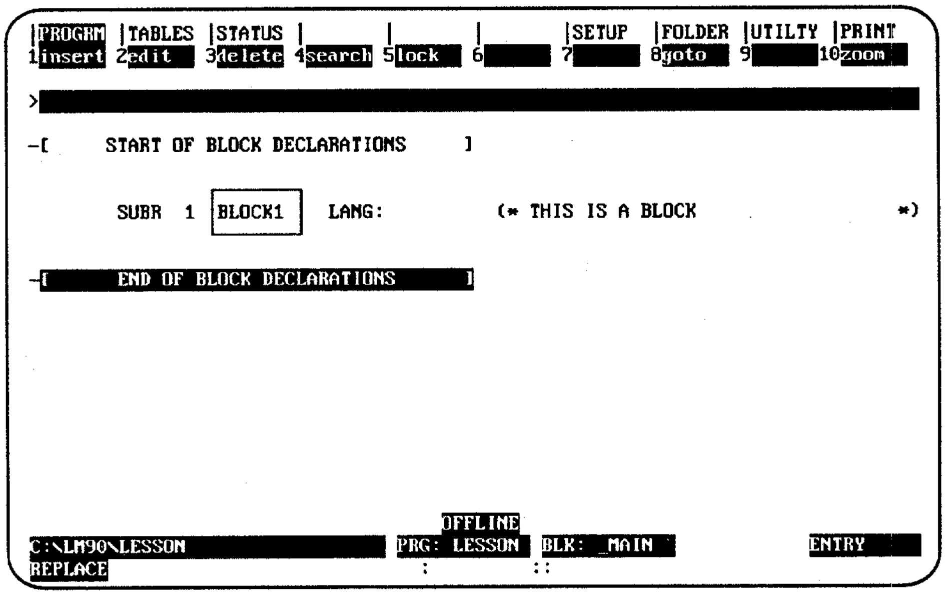
```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3delete 4search 5sort   6       7       8goto   9region10switch

>
       V A R I A B L E   D E C L A R A T I O N   T A B L E              Intake
                                                                        Valve
    REFERENCE   NICKNAME              REFERENCE DESCRIPTION             Control
    ---------   --------          -----------------------------         Switch
     %I0001      DWELL_T          weld dwell timer
     %I0033      LIMIT_T
     %I0037      FREE             spare input; not wired
     %I0101      I140_00
     %I0102      I140_01          Air Intake Valve Switch
     %I0105      I141_07          Intake Valve Control Switch
     %Q0202      START            machine start coil
     %M0627      B152_00          Bulb B152 Circ 00 Switch
     %M0997                       Sump Pump 1 Density Meter
     %T0086                       ConveyrCheck
     %R0111      REG_111          Registr Input Data
     %R0112      REG_112          Pointer to Read Data

                                       OFFLINE
C:\LM90\LESSON                        PRG: LESSON  BLK: _MAIN          ENTRY 0010
REPLACE                                  :          : :
```

This page is intentionally left blank.

Section 8: Subroutine Blocks

Subroutine blocks in Logicmaster 90-30/20/Micro software provide structured programming for the Series 90-30 PLC. Subroutine blocks are *not* available for the Series 90-20 PLC nor for Micro PLCs. For 90-30 PLCs, up to 64 subroutine block declarations are allowed.

Subroutines are declared through the block declaration editor. To create or modify subroutine declarations, place the cursor on the **[BLOCK DECLARATIONS]** marker. Then, press **Zoom** (**F10**) to display the block declarations screen.

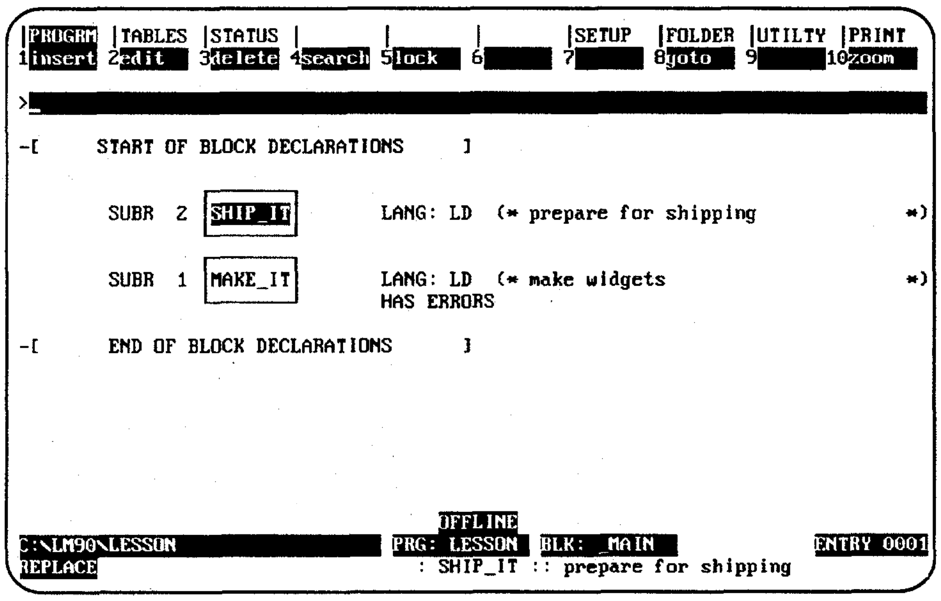


```
|PROGRM |TABLES |STATUS |        |        |        |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3delete 4search 5lock   6       7       8goto   9       10zoom
>

-[     START OF BLOCK DECLARATIONS      ]

        SUBR  1 |BLOCK1|  LANG:         (* THIS IS A BLOCK                  *)

-[      END OF BLOCK DECLARATIONS       ]



                                         OFFLINE
C:\LM90\LESSON                          PRG: LESSON  BLK: _MAIN           ENTRY
REPLACE                                  :            : :
```

Subroutine block declarations consist of:

- A label identifying the block as being used with a subroutine instruction.
- The number corresponding to the Hand Held Programmer's subroutine number.
- A graphic box containing a subroutine name of up to 7 alphanumeric characters.
- The language the subroutine was programmed in. Initially, this field is blank. It will remain empty until you have entered some logic for the subroutine. Once you have zoomed into the block to enter logic, the letters **LD** (ladder diagram language) are displayed after **LANG:** beside the block name.
- A 32-character descriptive explanation.
- An indicator that it has errors. This indicator is only displayed if the block is not executable. (See screen capture at top of the next page for an example of the error message.)

The following screen shows an example of two subroutine block declarations, one without errors and one with errors.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3delete 4search 5lock   6       7       8goto   9       10zoom
>

-[     START OF BLOCK DECLARATIONS      ]

        SUBR  2  SHIP_IT        LANG: LD  (* prepare for shipping              *)

        SUBR  1  MAKE_IT        LANG: LD  (* make widgets                      *)
                                HAS ERRORS

-[      END OF BLOCK DECLARATIONS       ]



                                         OFFLINE
C:\LM90\LESSON                          PRG: LESSON  BLK: _MAIN        ENTRY 0001
REPLACE                                   : SHIP_IT :: prepare for shipping
```

Adding Subroutine Block Declarations

To add a subroutine block declaration, place the cursor at the desired location, enter the block name on the command line, and press **Insert** (**F1**). A 32-character explanation can also be inserted at this time. If you press **F1** with the command line blank, a box containing seven question marks (???????) will be displayed for the subroutine block.

When the subroutine block declaration is inserted in the software, the next available Hand Held Programmer number (from 1 to 64) is assigned to it.

Editing Subroutine Block Declarations

To change a subroutine block declaration, place the cursor at the declaration to be changed and press **Edit** (**F2**). Type over the entry as needed. After changing the entry, press the **Enter** key (or Plus (+) key on the numeric keypad) to move to the next table entry and continue editing; or press the **Escape** key to exit editing.

Use the **Tab, Back Tab, Previous**, or **Next** key, or the cursor keys to move the cursor. Use the **Page Up** and **Page Down** keys to scroll the display up and down.

When you leave the subroutine block declarations, any changes made are automatically stored to the current program folder. You can also update the program folder while working on the screen by pressing **ALT-U**.

Deleting Subroutine Block Declarations

A subroutine block declaration can be deleted if there are no CALL instructions to that subroutine block in the program logic. Deleting a subroutine block also deletes its associated logic. To delete a subroutine block and declaration, place the cursor at the declaration to be deleted and press **Delete** (**F3**).

Searching for Subroutine Block Declarations

In order to search for a subroutine block declaration, you must have the Block Declaration screen displayed on your programmer. Then, press **Search** (**F4**). Enter the name of the subroutine block in the *Search for* field, set the *Scope* to LOCAL, and press the **Enter** key.

You can search for a subroutine CALL instruction from either the logic or the Block Declaration screen by pressing **Search** (**F4**). Then, enter the name of the subroutine block in the *Search for* field, or enter **&CALL** to search for all subroutine CALL instructions, and press the **Enter** key.

Using Goto

The **Goto** (**F8**) function key may be used to move the cursor. To move to a particular subroutine block declaration, enter the number of that declaration on the command line, and press **Goto** (**F8**). For example, to move the cursor to the first subroutine block declaration, enter `1` on the command line, and press **Goto** (**F8**).

You may also go to a subroutine block by simply entering the block name on the command line and pressing **Goto** (**F8**).

Zooming into Subroutine Block Logic

To display the subroutine block logic, place the cursor on the block name and press **Zoom** (**F10**). You can edit the logic on this screen.

Locking/Unlocking Subroutines

The block locking feature allows you to lock subroutines. Four types of locks are available:

Type of Lock	Description
View	Once locked, you cannot zoom into that subroutine.
Edit	Once locked, the information in the subroutine cannot be edited.
Perm View	The subroutine is permanently locked and cannot be unlocked.
Perm Edit	The subroutine is permanently locked and cannot be unlocked.

In addition to the locking capability, locked subroutines can also be unlocked, unless they are permanently locked.

A search or search and replace function may be performed on a view-locked subroutine. If the target of the search is found in a view-locked subroutine, one of the following messages is displayed, instead of logic:

For view-locked subroutines:

```
Found in locked block.   (Continue/Quit)
```

For edit-locked subroutines:

```
Cannot write to locked block.   (Continue/Quit)
```

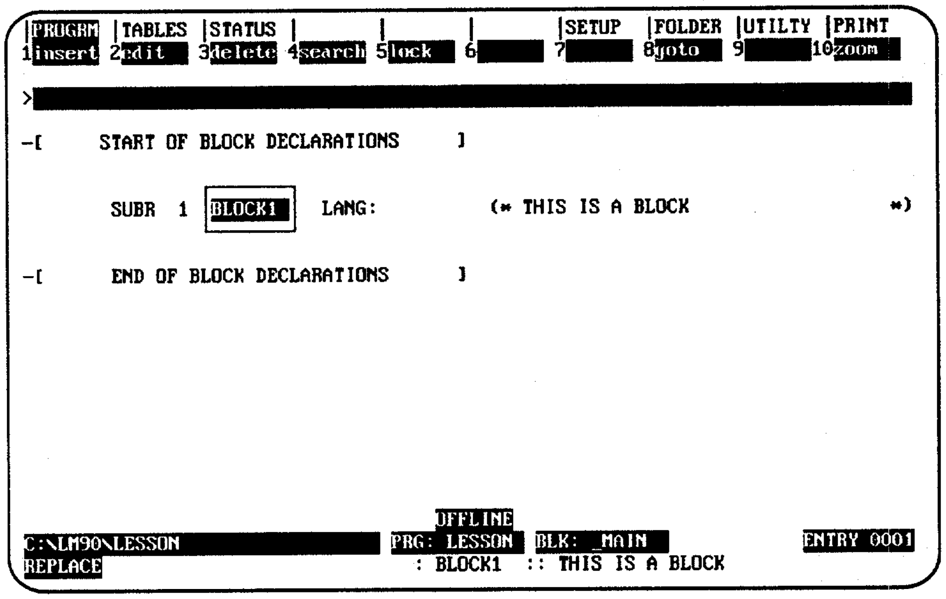
You may continue or abort the search. If you decide to continue, the locked subroutine is skipped and the search continues from the next subroutine. If you decide to quit, the search is aborted. For more information on search and search/replace, refer to chapter 3, section 11, "Search Function."

Folders that contain locked subroutines may be cleared or deleted. If a folder contains locked subroutines, these blocks remain locked when the Logicmaster 90-30/20/Micro software copy, backup, and restore folder functions are used. For more information on program folders, refer to chapter 7, "Program Folders."

Locking a Subroutine

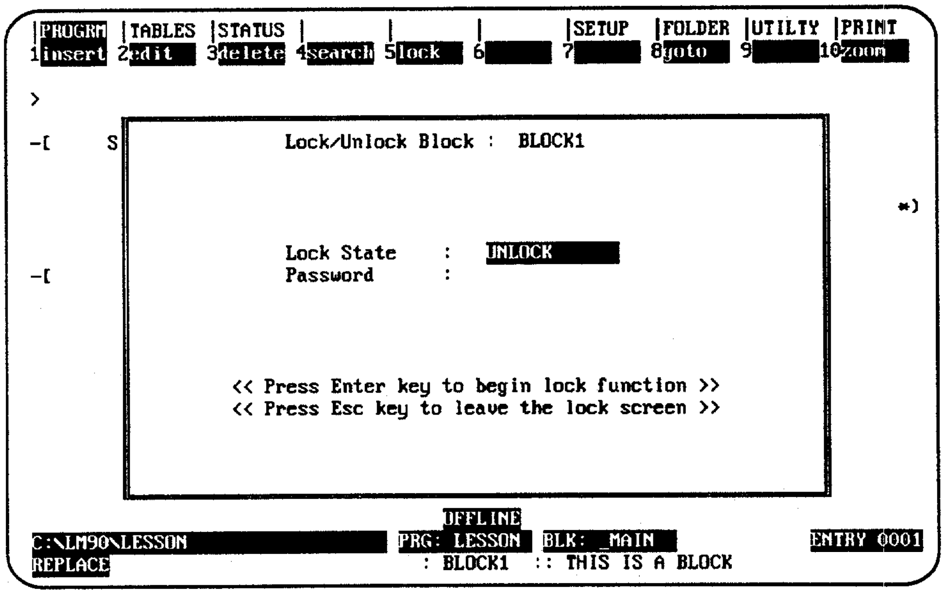
To lock a subroutine:

1. Move the cursor to the `[ BLOCK DECLARATIONS ]` marker and press **Zoom** (**F10**).



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3delete 4search 5lock   6       7       8goto   9       10zoom
>
-[      START OF BLOCK DECLARATIONS      ]

         SUBR  1  BLOCK1  LANG:          (* THIS IS A BLOCK                  *)

-[       END OF BLOCK DECLARATIONS       ]

                                          OFFLINE
C:\LM90\LESSON                            PRG: LESSON  BLK: _MAIN       ENTRY 0001
REPLACE                                     : BLOCK1  :: THIS IS A BLOCK
```

2. In the block declaration editor, move the cursor to the desired subroutine and press Lock (F5) to display the Lock/Unlock Block screen. The subroutine name is included as part of the title on this screen.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3delete 4search 5lock   6       7       8goto   9       10zoom
>
-[     S            Lock/Unlock Block :  BLOCK1
                                                                           *)

                    Lock State    :  UNLOCK
-[                  Password      :


                << Press Enter key to begin lock function >>
                << Press Esc key to leave the lock screen >>

                                          OFFLINE
C:\LM90\LESSON                            PRG: LESSON  BLK: _MAIN       ENTRY 0001
REPLACE                                     : BLOCK1  :: THIS IS A BLOCK
```

Field	Description
Lock State	Specify the operation to be performed on the current subroutine. Use the **Tab** key to view the values for this field: • UNLOCK: The current subroutine is unlocked. • VIEWLOCK: A view lock is set on the subroutine, and you cannot zoom into the subroutine. If you try to zoom into the subroutine, the error message "Zoom denied because the block is locked" is displayed. • EDITLOCK: An edit lock is set on the subroutine, and the subroutine cannot be changed. If you try to edit the subroutine, the error message "Edit denied because the block is locked" is displayed. • PERMVIEWLOCK or PERMEDITLOCK: The subroutine is **permanently** locked and cannot be unlocked once it is locked. When the desired value is displayed in the *Lock State* field, press the **Enter** key. If you try to enter an incorrect value or leave the field empty, a message is displayed in the message area of the screen and the field remains active. If you try to lock a subroutine that is already locked, the message "Block already locked" is displayed in the message area and the screen remains displayed.
Password	Specify a password of up to four characters to lock a subroutine. Valid characters include A through F and 0 through 9. If a lowercase letter is entered, it is converted to an uppercase letter. Once a password is set, the same password must be entered before the subroutine can be unlocked. The characters of the password are displayed as they are typed into the *Password* field. For **PERMVIEWLOCK** and **PERMEDITLOCK**, the *Password* field is ignored. Passwords are not required and cannot be used with permanently locked blocks.

3. If no lock is set on the current subroutine, the *Lock State* field is set to **UNLOCK** when the screen is displayed. If a lock is set on the current subroutine, then the *Lock State* field is set to the type of lock imposed on the block. Initially, the *Password* field is empty.

4. To set the lock, enter the value in the *Lock State* field. Move the cursor to the *Password* field, and enter a password. Then, press the **Enter** key. The software will prompt you to confirm the locking process with the message "Is logic block to be locked? (Y/N)".

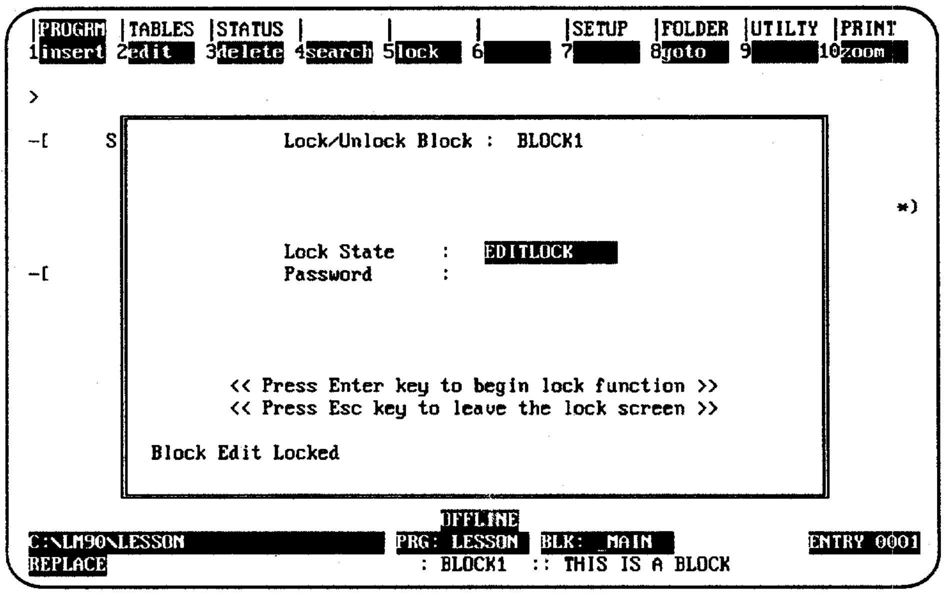
5. If you enter **Y** (Yes) after the confirmation prompt, the locking process begins. Entering **N** (No) aborts the process. Once the subroutine is successfully locked, the message "Block locked" is displayed in the message area of the screen, the type of lock and password are written to the subroutine lock header, and the *Password* field is cleared.

6. To quit the Lock/Unlock Block screen, press the **Escape** key. To restore the value of the fields to their original values, press **ALT-A**.

Unlocking a Subroutine

A previously view-locked or edit-locked subroutine may be unlocked in the block declaration editor, unless it is permanently view locked or permanently edit locked.

1. Move the cursor to the desired subroutine block and press **Lock** (**F5**) to display the Lock/Unlock Block screen.



```
|PROGRM |TABLES |STATUS |        |        |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3delete 4search 5lock    6       7       8goto   9       10zoom
>
-[      S     Lock/Unlock Block :  BLOCK1
                                                                          *)

              Lock State    :  EDITLOCK
-[            Password      :

          << Press Enter key to begin lock function >>
          << Press Esc key to leave the lock screen >>

        Block Edit Locked
                              OFFLINE
C:\LM90\LESSON                PRG: LESSON  BLK: MAIN              ENTRY 0001
REPLACE                       : BLOCK1  :: THIS IS A BLOCK
```

The current subroutine name is included as part of the title on this screen. If the current subroutine is locked, the *Lock State* field indicates the type of lock. If the subroutine is not locked, the field is set to **UNLOCK**. The *Password* field is initially empty.

2. To unlock the subroutine, set the value of the *Lock State* field to **UNLOCK**.

3. Move the cursor to the *Password* field, and enter the correct password. As each character of the password is typed, an asterisk is displayed in the field.

4. Then, press the **Enter** key. The password you entered is compared with the password last set. If the two passwords are identical, the subroutine is unlocked and the message "Block unlocked" is displayed in the message area of the screen. Once the subroutine is unlocked, the *Lock State* field is set to **UNLOCK** and the *Password* field contains the password of the block just unlocked. In addition, the password is cleared in the subroutine header.

5. If the passwords do not match, the subroutine is not unlocked, the error message "Incorrect password given for unlock" is displayed in the message area of the screen, the *Password* field is cleared, and the Lock/Unlock Block screen remains displayed.

6. To quit the Lock/Unlock Block screen, press the **Escape** key. To restore the value of the fields to their original values, press **ALT-A**.

Permanently Locking a Subroutine

In addition to **VIEWLOCK** and **EDITLOCK**, there are two types of permanent locks. If a **PERMVIEWLOCK** lock is set, all zooms into a subroutine are denied. If a **PERMEDITLOCK** lock is set, all attempts to edit the block are denied. Therefore, passwords are not required and cannot be used with permanently locked subroutines.

Warning

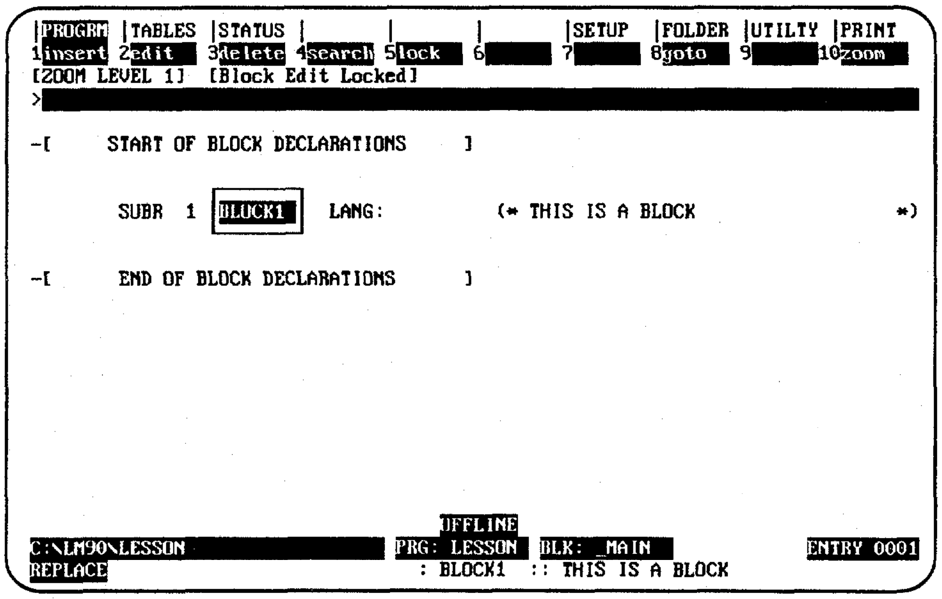
Permanent locks differ from the regular VIEWLOCK and EDITLOCK in that once set, they cannot be removed.

Once a **PERMEDITLOCK** is set, it can only be changed to a **PERMVIEWLOCK**; you cannot unlock the block first and then set a **PERMVIEWLOCK**. A **PERMVIEWLOCK** cannot be changed to any other type of lock.

When you press the **Enter** key to initiate the locking, the software will prompt you to confirm any permanent lock.

Displaying the Lock Status of the Subroutine

The display zoom level function (ALT-X) can be used to display the lock status of the subroutine in the block declaration editor. Move the cursor to the desired block, and press **ALT-X**. The following example screen shows a block that is locked for view.



```
|PROGRM |TABLES |STATUS |        |        |        |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3delete 4search 5lock   6       7       8goto   9       10zoom
[ZOOM LEVEL 1]  [Block Edit Locked]
>

-[     START OF BLOCK DECLARATIONS      ]

        SUBR  1  BLOCK1   LANG:          (* THIS IS A BLOCK                    *)

-[     END OF BLOCK DECLARATIONS        ]



                                              OFFLINE
C:\LM90\LESSON                                PRG: LESSON  BLK: _MAIN          ENTRY 0001
REPLACE                                         : BLOCK1  :: THIS IS A BLOCK
```

Periodic Subroutines

Model 341 CPUs, Version 4.20 or later, and Model 340 CPUs support the use of a periodic subroutine. A periodic subroutine is a single subroutine with a unique name in the form **1Tiiii**, which will execute periodically during **RUN** mode. If a subroutine with this name is present in the PLC when it transitions from **STOP** to **RUN** mode, this subroutine will be executed at periodic intervals while the PLC is in **RUN** mode. If a **RUN MODE STORE** is performed, all periodic subroutines will be stopped while the PLC is in **PAUSE** mode. (For more information on **RUN MODE STORE** and **PAUSE** mode, refer to chapter 8, "Program Utilities.")

Note

Only Model 340 and higher Series 90™-30 PLCs support use of a periodic subroutine.

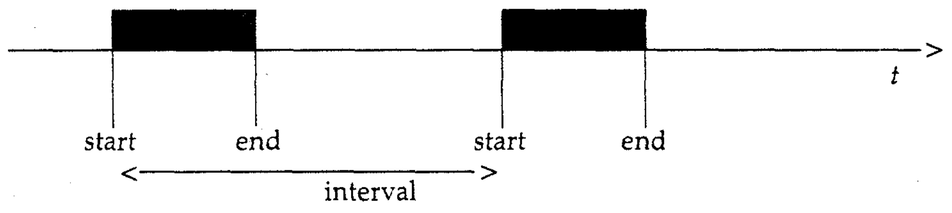
You can specify a periodic subroutine by giving the block a name in the form: **1T0001**, where the four digits after the **T** indicate the number of time intervals between executions of the block. If the leading zeros of the interval are not specified, the Logicmaster 90-30/20/Micro software will fill them in so that the total number of characters is six. If the time interval specified is too small for execution of the subroutine and the rest of the normal PLC sweep, the PLC watchdog timer may be activated. The time per interval is .001 seconds. The maximum time allowed between executions is 10 milliseconds.

When you enter a subroutine name with the correct format for a periodic subroutine, the timebase, interval value, and "INTR" will be displayed beside the subroutine's name and number, instead of "SUBR."

The PLC will look for periodic subroutines when a declaration block is stored, when the program is cleared, and when a subroutine is deleted.

Executing a Periodic Subroutine

Each execution of the periodic subroutine will occur interval seconds after the previous start, as shown below:



Note

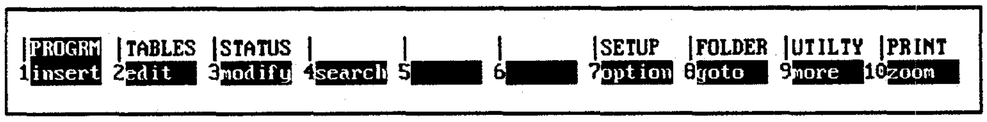
The latency for the periodic subroutine (i.e., the maximum interval between the time the periodic subroutine should have executed and the time it actually executes) can be around .35 milliseconds if there is no PCM, CMM, or ADC module in the main rack. If there is a PCM, CMM or ADC module in the main rack — even if it is not configured or used — the latency can be almost 2.25 milliseconds. For that reason, use of the periodic subroutine with PCM-based products is not recommended.

Restrictions on Use of the Periodic Subroutine

- Periodic subroutines cannot be called by the main program or by another subroutine.
- If you attempt to load a program with a periodic subroutine into a version of Logicmaster 90-30 software prior to Release 4.01, you will not be able to display or edit the program. Likewise, you cannot store a periodic subroutine to a PLC prior to Version 4.20.
- If a DOIO function block whose I/O reference range includes an intelligent module cannot be executed within a periodic subroutine, communication with the module may be lost.
- Time (TMR, ONDTR, and OFDTR) function blocks will not execute properly within a periodic subroutine. A DOIO function block within a periodic subroutine whose reference range includes references assigned to a Smart I/O Module (HSC, APM, Genius, etc.) will cause the CPU to lose communication with the module. The FST_SCN and LST_SCN contacts (%S1 and %S2) will have an indeterminate value during execution of the periodic subroutine. A periodic subroutine cannot call or be called by other subroutines.

Section 9: Rung Edit

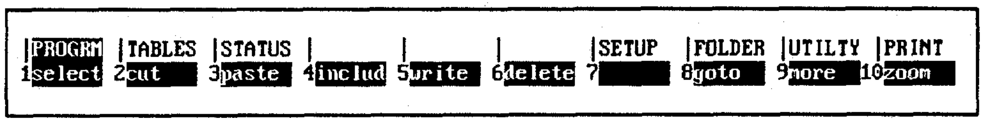
After a logic program has been entered, there is always a need to make modifications, either to correct logic errors or to add new capabilities. Logicmaster 90-30/20/Micro software has an array of features to make this process easy. The same function keys used to initially enter a rung are available to insert new rungs. In addition, there are function keys to edit or modify existing rungs.



```
|PROGRM |TABLES |STATUS |        |        |        |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3modify 4search 5       6       7option 8goto   9more  10zoom
```

Function Key	Function	Description
F1	Insert	Add one or more rungs to the program. Selecting the insert function opens a new space above the rung where the cursor is positioned and displays the Edit Rung keys, which are used to select program elements. This allows you to create a new rung.
F2	Edit	Edit the rung at the cursor location. The edit function activates the current rung, enabling you to modify that rung. When **F2** is pressed, the Edit Rung keys are displayed at the top of the screen. These keys are used to select program elements.
F3	Modify	Initiate word-for-word instruction changes using the rung edit softkeys instead of mnemonics.
F4	Search	Locate a program element.
F7	Option	Access coil checking and other editor options.
F8	Goto	Go to the specified rung.
F9	More	Access additional rung edit softkeys.
F10	Zoom	Go to a more detailed level. To return to the original level, press the **Escape** key.

Pressing **More** (**F9**) displays these additional rung edit softkeys.



|PROGRM |TABLES |STATUS | | | |SETUP |FOLDER |UTILTY |PRINT
1select 2cut 3paste 4includ 5write 6delete 7 8goto 9more 10zoom

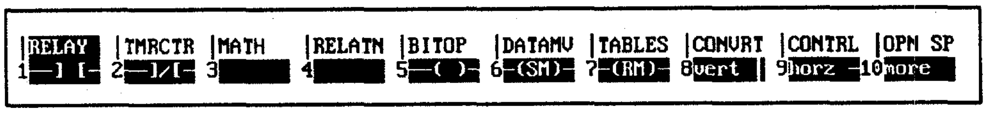
Function Key	Function	Description
F1	Select	Select or deselect a rung or group of rungs. Rungs or declarations may also be deselected by pressing **F1** again, by pressing **ALT-A**, or by pressing the **Escape** key. After deselecting the rungs or declarations, the message "Select mode cancelled" is displayed on the message line and the cursor remains on the last rung or declaration that had been selected.
F2	Cut	Delete the selected section of rungs or declarations from a program and put them in the cut buffer. After a cut operation, the cursor will appear on the rung or declaration immediately after the selected region.
F3	Paste	Insert previously cut rungs or declarations from the cut buffer. Cut rungs and declarations may be pasted any number of times, until they are replaced by new information in the cut buffer, or until the program editor is exited.
F4	Include	Insert previously written rungs or declarations back into the original program, or into any other program. Written rungs and declarations may be included any number of times.
F5	Write	Copy the selected section of rungs or declarations from a program into a special file. After a write, the cursor remains on the last rung or declaration selected.
F6	Delete	Delete the rung at the cursor location or a range of selected rungs.
F8	Goto	Enter the rung number on the command line, and then press **F8** to "go to" a specific rung in the ladder diagram logic. You can also specify a subroutine number and rung number in the subroutine to "go to" a specific rung in that subroutine.
F9	More	Return to the first level of rung edit softkeys.
F10	Zoom	Zoom into the item the cursor is on.

Note

The function keys listed above can also be applied to the variable declaration table. Using the file write function, a group of reference descriptions can be written to a file to be used by another program.

Editing a Rung

When either the insert, edit, or modify function is activated by pressing **F1, F2, or F3**, respectively, the following softkey selections are displayed at the top of the screen.



```
|RELAY  |TMRCTR |MATH    |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1—] [- 2—]/[- 3        4        5—( )- 6-(SM)- 7-(RM)- 8vert  | 9horz -10more
```

These function keys provide access to the instructions required to edit ladder diagram rungs.

Note

Only one rung is active at a time in either the insert or edit function. Each rung must be completed <u>and</u> accepted by the software before the next rung can be edited.

Entering Insert or Edit Mode

Enter either **INSERT** or **EDIT** mode by:

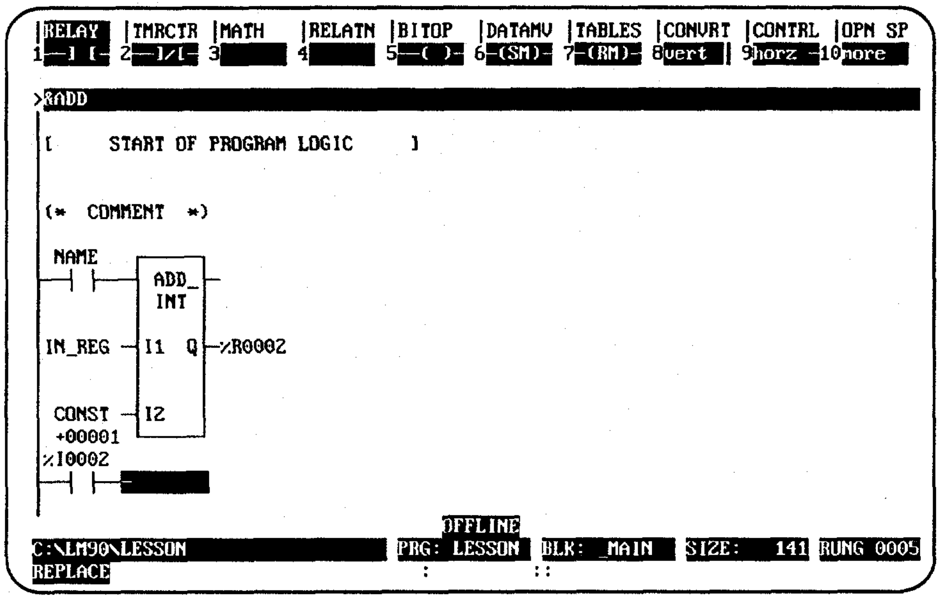
1. Pressing **F1** to enter **INSERT** mode. In this mode, the new rung is inserted before the rung on which the cursor is positioned. Therefore, make sure the cursor is on the rung following the location for the new rung before you press **F1**.
2. Pressing **F2** to enter **EDIT** mode. Then, move the cursor to the rung element you want to change.

Entering Instructions

Select instructions by:

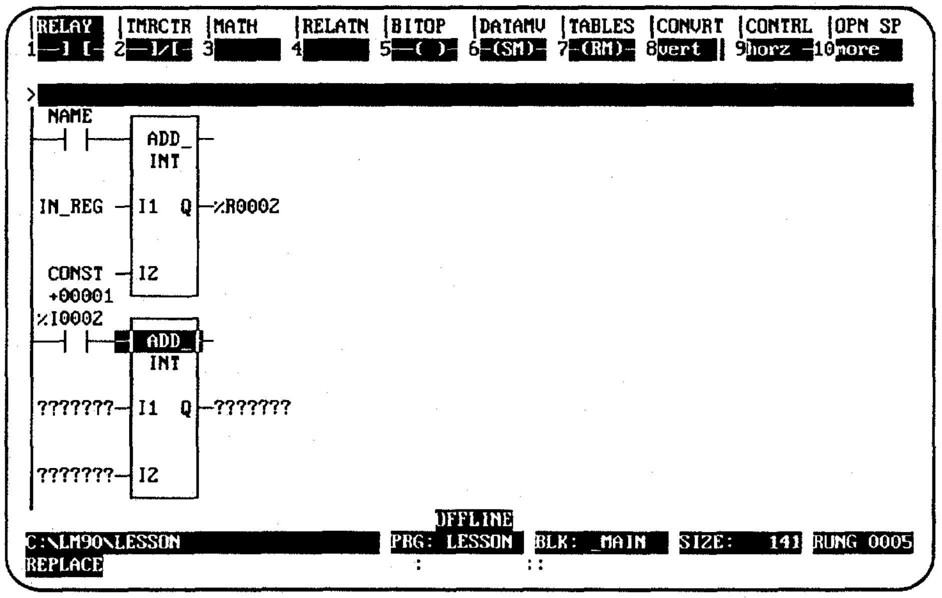
1. *After entering Insert or Edit mode* (as discussed above), typing the mnemonic for the instruction, preceded by an ampersand character (&). (Refer to appendix D, *Instruction Mnemonics,* for a listing of the mnemonics for Logicmaster 90-30/20/Micro programming software.) Then, press the **Enter** key.
2. *After entering Insert or Edit mode* (as discussed above), pressing the **Shift** key and a function key to display a specific group of instructions. Then, select a specific instruction within that group by pressing its function key.

In this example, the ADD function is selected by typing the mnemonic: **&ADD** on the command line.



```
|RELAY |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1-] [- 2-]/[- 3       4       5-( )- 6-(SM)- 7-(RM)- 8vert  | 9horz -10more
>&ADD
[     START OF PROGRAM LOGIC     ]

(*  COMMENT  *)

 NAME
-| |----| ADD_ |-
        | INT  |

IN_REG -|I1   Q|-%R0002

 CONST -|I2    |
 +00001
%I0002
-| |-

                                    OFFLINE
C:\LM90\LESSON                      PRG: LESSON BLK: _MAIN  SIZE:   141 RUNG 0005
REPLACE                                 :          ::
```

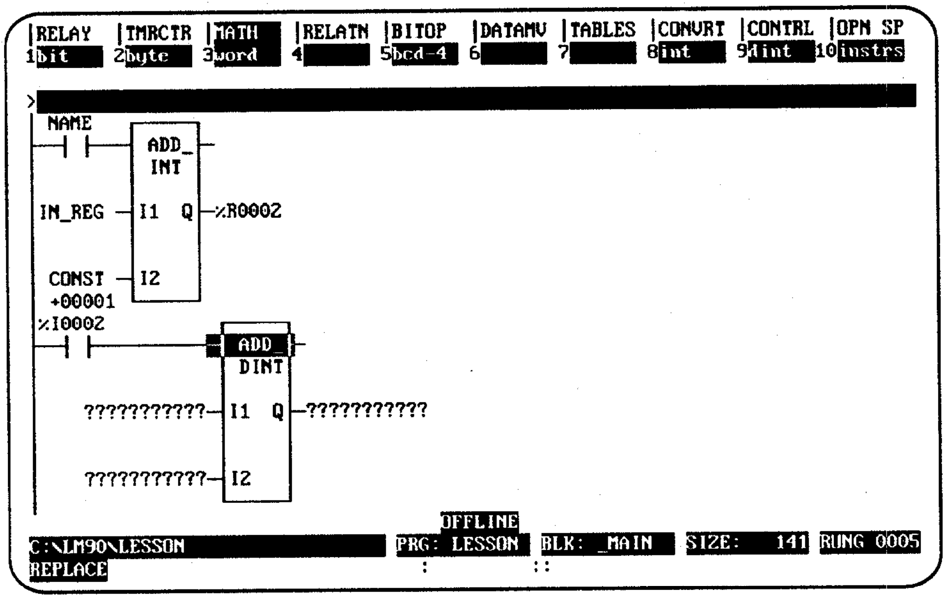
After pressing the `Enter` key, the screen adjusts to display the complete new instruction and the ADD function is inserted.



```
|RELAY |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1-] [- 2-]/[- 3       4       5-( )- 6-(SM)- 7-(RM)- 8vert  | 9horz -10more
>
 NAME
-| |----| ADD_ |-
        | INT  |

IN_REG -|I1   Q|-%R0002

 CONST -|I2    |
 +00001
%I0002
-| |----| ADD_ |-
        | INT  |

???????-|I1   Q|-???????

???????-|I2    |

                                    OFFLINE
C:\LM90\LESSON                      PRG: LESSON BLK: _MAIN  SIZE:   141 RUNG 0005
REPLACE                                 :          ::
```

Entering/Modifying Data Types

Some instructions support different data types. To change ADD_INT to ADD_DINT, press **`Types`** (**`F10`**) with the math functions still displayed on the screen. Then, press **`DINT`** (**`F9`**) with the cursor on the ADD function block to create the following screen. An alternative way to select ADD_DINT is to type **`&ADD_DINT`** on the command line, and then press the **`Enter`** key.



```
|RELAY  |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1bit    2byte   3word   4       5bcd-4  6       7       8int    9dint  10instrs
>
 NAME
 -| |---[ADD_INT]-
 IN_REG -I1  Q-%R0002
 CONST  -I2
 +00001
%I0002
 -| |---------[ADD_DINT]-
  ???????????-I1  Q-???????????
  ???????????-I2
                                        OFFLINE
C:\LM90\LESSON                          PRG: LESSON  BLK: _MAIN   SIZE:   141 RUNG 0005
REPLACE                                     :          ::
```

Moving the Cursor within a Rung

Use the keys listed in the table below to move within the rung.

Key	Description
Cursor keys Tab and Shift-Tab Home key End key	Move the cursor within the rung. Move among the inputs and outputs of a function. Position the cursor on column 1 in row 1. Position the cursor on the last defined row.
For Command Line Entry	
Insert key Delete key Backspace key CTRL-Left Cursor or CTRL-Right Cursor	Change the text editing mode (**`INSERT`** or **`REPLACE`**). Delete the character at the cursor position. Delete the character to the left of the cursor position. Move the cursor within the command line.

Entering a Reference Address

Enter a reference address on the command line. Then, press the **Enter** key to apply the reference to the operand at the current cursor location. Repeat this step for each operand in the instruction. For multiple operand instructions, you may find that pressing the **Tab** key is more efficient because doing so causes Logicmaster to apply the reference to the operand at the current cursor location *and* automatically moves the cursor to the next operand location awaiting your input.

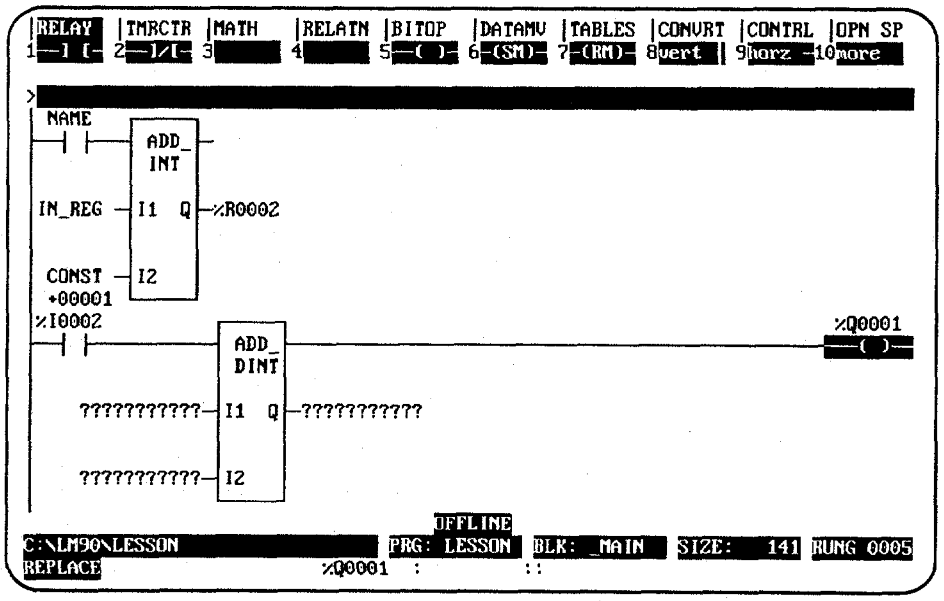
For single operand instructions, the reference address can be entered on the command line before pressing the function key (described in the previous step). Then, when the function key is pressed, the reference address is automatically applied to the instruction.

Entering Nicknames

Nicknames can also be created or modified on the command line. To do this, enter the nickname together with its machine reference and any associated reference description. (Refer to chapter 3, section 4, "Program Annotation," for more information on entering nicknames and reference descriptions.)

Using Vertical and Horizontal Links

Use **Vertical Links (F8)** and **Horizontal Links (F9)** to connect the instructions within a rung. These links are available from the relay functions, but may also be entered without returning to the relay function menu by pressing the Vertical bar (|) key to enter a vertical link or the Tilde (~) key to enter a horizontal link.

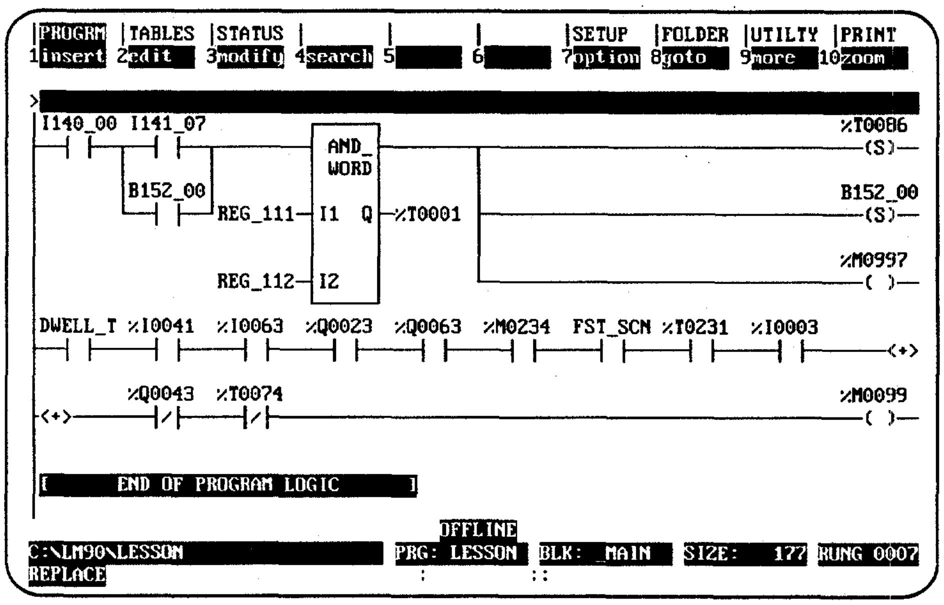


Using Continuation Coils and Contacts

Use the **Continuation Coil** (**F1**) and the **Continuation Contact (F2)** to continue relay ladder rung logic beyond the limit of ten columns. These keys are available from the relay functions by pressing **More** (**F10**); they may also be entered using their mnemonics (&COILCTD and &CONCTD). There can be only one continuation coil and/or one continuation contact per rung.

The **Continuation Coil (F1)** is used to carry the current rung's status to the rung which has a continuation contact. The continuation coil can only be placed in column 10 of the rung. It does not require the use of a continuation contact in following rungs for editing.

The **Continuation Contact** (**F2**) is used to continue the status of the continuation coil rung's logic on the continuation contact's rung.



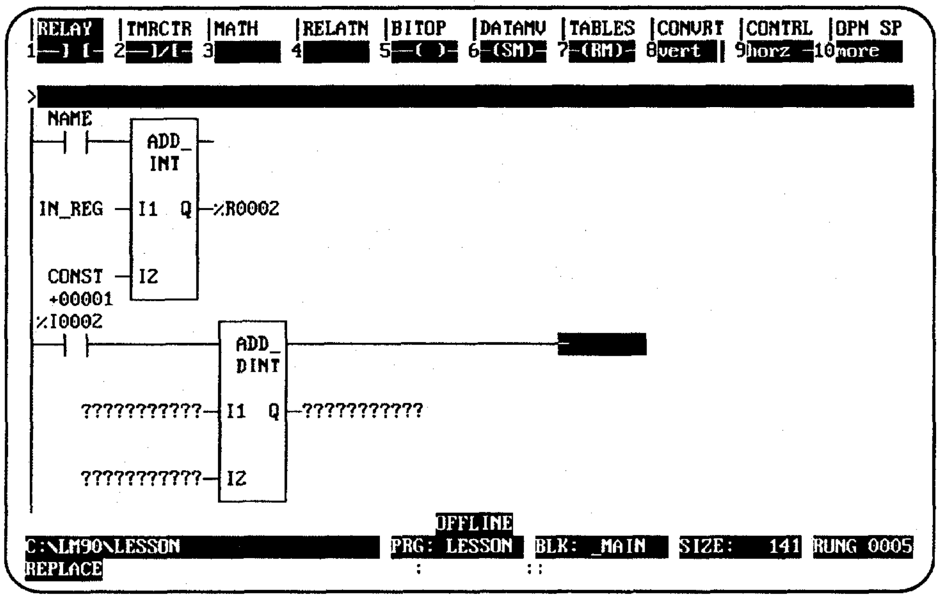
```
|PROGRM |TABLES |STATUS |        |       |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3modify 4search  5       6       7option 8goto   9more   10zoom

>
|I140_00 I141_07                                                     %T0086
|--| |----| |--+        +------+   +-------------------------------------(S)--
|              |        | AND_ |   |
|              |        | WORD |   |
|  B152_00     |        |      |   |                                 B152_00
|   +--| |----+REG_111-| I1  Q|-%T0001 +-----------------------------------(S)--
|                       |      |   |
|                       |      |   |                                 %M0997
|              REG_112-| I2   |   +-------------------------------------( )--
|                       +------+
|DWELL_T %I0041  %I0063  %Q0023  %Q0063  %M0234  FST_SCN %T0231  %I0003
|--| |-----| |-----| |-----| |-----| |-----| |-----| |-----| |-----| |--------<+>
|
|        %Q0043  %T0074                                              %M0099
|<+>--------|/|-----|/|-------------------------------------------------( )--
|
|[        END OF PROGRAM LOGIC        ]
|
                                   OFFLINE
C:\LM90\LESSON                     PRG: LESSON  BLK: _MAIN  SIZE:    177 RUNG 0007
REPLACE                                 :           ::
```

The state of the last executed continuation coil is the flow state that will be used on the next executed continuation contact. The continuation contact does not require the use of a continuation coil in previous rungs for editing. However, if the flow of logic does not execute a continuation coil before it executes a continuation contact, the state of that contact will be no flow.

Deleting an Element

To delete an operand or instruction from a rung while in **INSERT** or **EDIT** mode, press **ALT-D** or **Open Space** (**Shift-F10**) and then press **Delete Instruction** (**F10**). Enter another instruction in the space left by the deleted instruction, or enter a horizontal link. An instruction may also be deleted by replacing it with another instruction.

To remove the horizontal links and coil that follow the ADD_DINT instruction, place the cursor on each logic segment and press **ALT-D**. The **Tilde** (~) key or **Horizontal Link** (**F9**) softkey can also be used to clear the link.



```
|RELAY  |TMRCTR |MATH    |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1-] [-  2-]/[-  3        4       5-( )-  6-(SM)- 7-(RM)- 8vert | 9horz -10more
```

```
                                        OFFLINE
C:\LM90\LESSON                          PRG: LESSON  BLK: _MAIN  SIZE:   141 RUNG 0005
REPLACE
```

Using Open Space Functions

To add open space to a rung in **INSERT** or **EDIT** mode, select **Open Space** (**Shift-F10**). Then, select the specific function key described below:

Function Key	Function	Description	Page
F1	Move Right	Move the element at the cursor position, and all the elements to the right of the cursor position, one position to the right in all lines. Shunts are automatically inserted into the new column in the rung.	3-85
F3	Move Down	Create room for an additional line of logic above the line the cursor is on.	3-86
F5	Delete Column	Delete an entire column of instructions (including verticals) and operands within a rung. All elements to the right of the deleted column will automatically move left.	3-87
F7	Delete Row	Delete an entire row of instructions (including verticals) and operands within a rung.	3-88
F9	Delete Nickname	Remove nicknames from the variable declaration table. Enter the nickname to be deleted on the command line before pressing **F9**.	N/A
F10	Delete Instruction	Delete an operand or instruction from a rung. (You may also press **ALT-D** to delete an operand or instruction.)	3-89

Completing (Accepting) Rung Entry

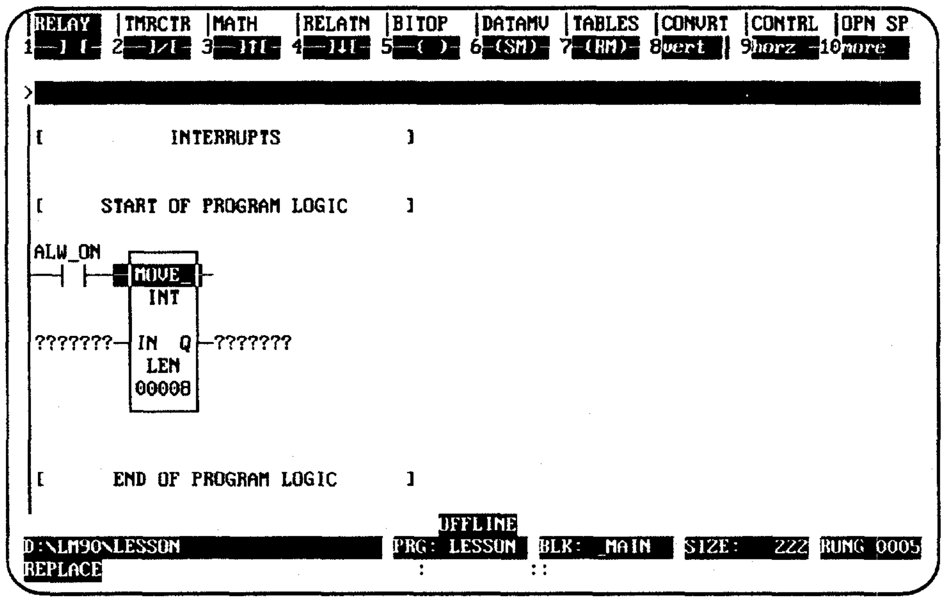
After changing the rung, press the **Enter** key with the command line empty (or **Plus** (+) key on the numeric keypad) to accept the edited rung and move the cursor to the next rung for editing. In **EDIT** mode, you must press the Edit (F2) key after the rung is accepted in order to edit the next rung. In **INSERT** mode, it is not necessary to press **Edit** (**F2**). You will still be in **INSERT** mode after the rung is accepted and can enter the next rung. You will remain in **INSERT** mode until you press the **Escape** key.

Press **ALT-A** to quit the rung without saving any changes in the program.

Entering an Instruction Length

To change the length of the instruction (in this example to *8*), do the following:

- Position the cursor on the function.
- Type **8** (i.e., the length appropriate for your situation—*8* in this example) on the command line.
- Press the **Enter** key.

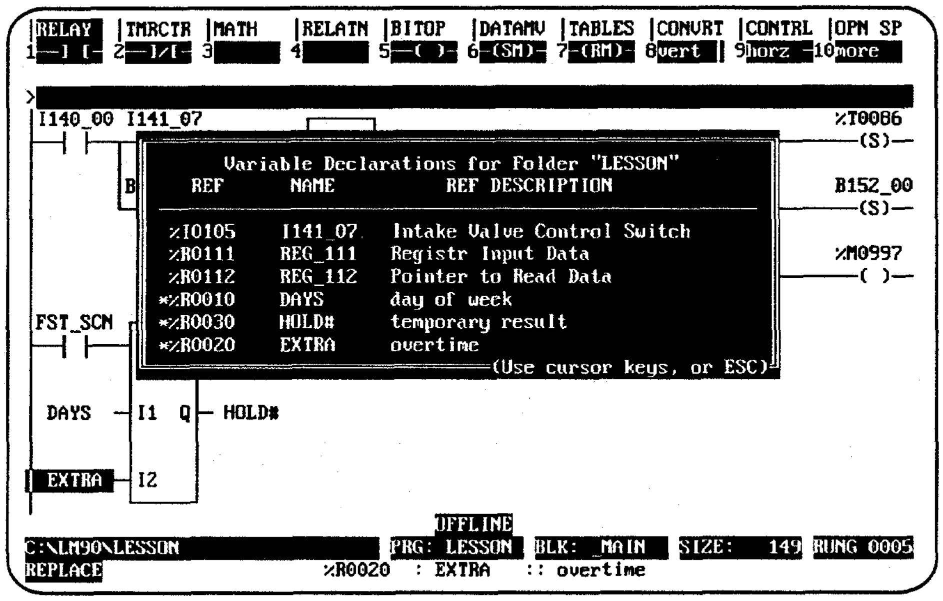


```
RELAY  |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1-] [- 2-]/[- 3-]T[- 4-]↓[- 5-( )- 6-(SM)- 7-(RM)- 8vert | 9horz -10more
>

[          INTERRUPTS          ]

[     START OF PROGRAM LOGIC   ]

ALW_ON
--| |--[MOVE_]-
        INT
???????-IN   Q-???????
        LEN
       00008

[      END OF PROGRAM LOGIC    ]
                                    OFFLINE
D:\LM90\LESSON                      PRG: LESSON  BLK: _MAIN  SIZE:   222  RUNG 0005
REPLACE
```

Viewing Variable Declarations

To view the current set of variable declarations from anywhere in the Program Editor, press **ALT-V** to invoke the variable declaration window. The window allows cursor and page key scrolling. Pressing **ALT-V** again will show the next table if it exists (local or global).

In **RUNG INSERT** or **EDIT** mode, an asterisk (*) is displayed next to the variable declarations that were created or modified during the current editing session. These new or modified declarations will remain declared if the rung is accepted.



```
|RELAY  |TMRCTR |MATH    |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1-] [- 2-]/[- 3        4       5-( )- 6-(SM)- 7-(RM)- 8vert | 9horz -10more

              Variable Declarations for Folder "LESSON"
     REF          NAME            REF DESCRIPTION

    %I0105      I141_07    Intake Valve Control Switch
    %R0111      REG_111    Registr Input Data
    %R0112      REG_112    Pointer to Read Data
   *%R0010      DAYS       day of week
   *%R0030      HOLD#      temporary result
   *%R0020      EXTRA      overtime
                           ═════(Use cursor keys, or ESC)

                                        OFFLINE
C:\LM90\LESSON                       PRG: LESSON  BLK: _MAIN  SIZE:  149 RUNG 0005
REPLACE                    %R0020  : EXTRA   :: overtime
```

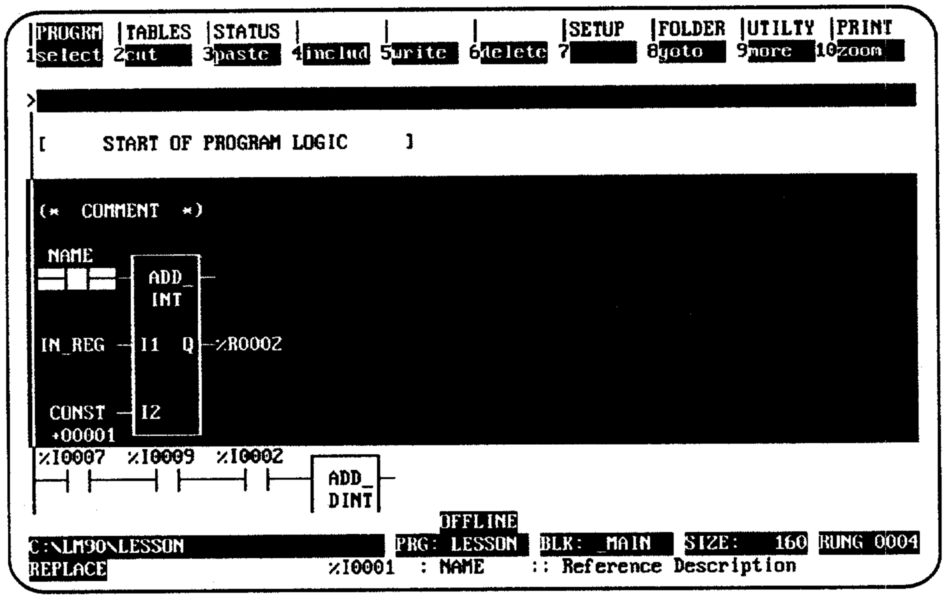
Deleting a Rung

To remove a single rung of logic, place the cursor at the rung, press **More** (**F9**), and then press **Delete** (**F6**); or simply press **ALT-D** (while not in **INSERT** or **EDIT** mode).

Selecting Rungs

The cutting and file writing of rungs requires that a section of rungs first be selected. To select the rungs:

1. Press **More** (**F9**) to display the cut/paste function keys.
2. Move the cursor to the rung at the beginning or end of the section to be selected. Then, press **Select** (**F1**). The message "Select mode initiated ..." displayed on the message line indicates that the select function has been activated. Once this function is active, the only functions available are cursor, page, cut, write, help, and goto.
3. The current rung is automatically selected. By pressing the Up/Down cursor keys, Prev, Next, Page Up/Down, Home, or End keys, you can add subsequent or previous rungs into the selected region. Selected rungs are shown in reverse video on the display.


```
|PROGRM  |TABLES |STATUS |        |        |        |SETUP  |FOLDER |UTILTY |PRINT
1select  2cut    3paste  4includ  5write   6delete  7       8goto   9more   10zoom
```

```
                                                OFFLINE
C:\LM90\LESSON                              PRG: LESSON  BLK: _MAIN  SIZE:   160 RUNG 0004
REPLACE                          %I0001  : NAME     :: Reference Description
```

4. The Select (F1) softkey acts as a toggle between the select and deselect functions. To deselect the selected rungs, press the **Select** (**F1**) key or **ALT-A**.
   A. Press **Cut** (**F2**) to cut the selected rungs.
   B. Press **Write** (**F5**) to write the selected rungs to a file.
   C. Press **Delete** (**F6**) or **ALT-D** to delete the selected rungs.
5. To deselect the selected rungs, press **Select** (**F1**). The **F1** key acts as a toggle between the select and deselect functions.

Note

The select function can also be used to select variable declarations and interrupt declarations.

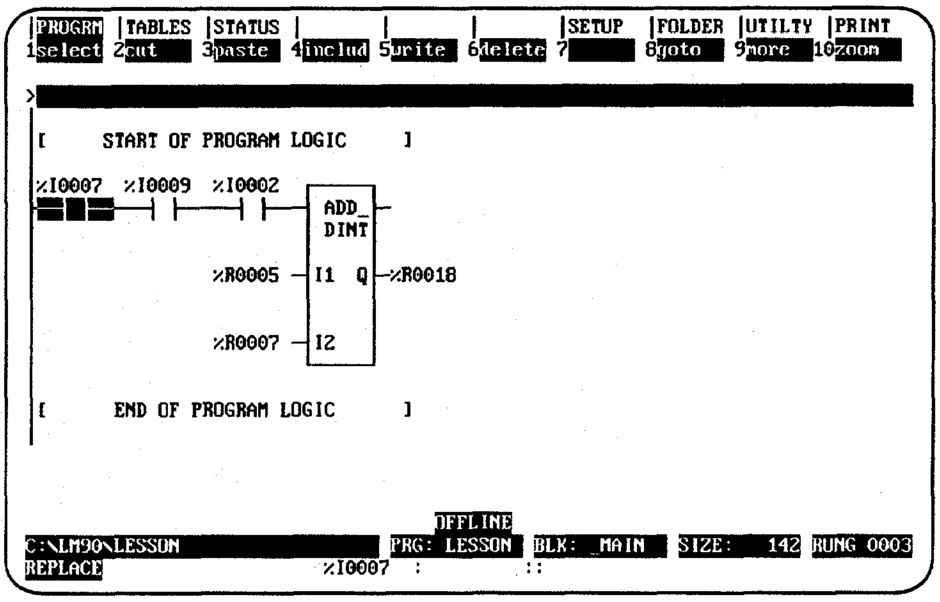
Cutting Selected Rungs

The cut function enables you to remove a section of previously selected rungs from the current program. Any nicknames, reference descriptions, or comment annotation used within the selected rungs are also copied with the cut rungs for later use in a paste operation. However, nicknames and reference descriptions remain in the variable declaration table.

This function can be used to reorder logic within the program; it is also useful for deleting a section of rungs.

To cut a section of rungs from the current program's logic:

1. Select one or more rungs, as previously described.
2. Press **Cut** (**F2**) to cut the selected rungs from the program. The select function is automatically exited as part of the cut operation. The cursor will be on the rung after the cut section of rungs.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1select 2cut    3paste  4includ 5write  6delete 7       8goto   9more   10zoom
>
[     START OF PROGRAM LOGIC     ]

%I0007  %I0009  %I0002
---| |-----| |-----| |----+-------+-
                          | ADD_  |
                          | DINT  |
                          |       |
                 %R0005 --|I1    Q|--%R0018
                          |       |
                          |       |
                 %R0007 --|I2     |
                          +-------+

[      END OF PROGRAM LOGIC      ]

                                     OFFLINE
C:\LM90\LESSON                       PRG: LESSON  BLK: _MAIN   SIZE:   142  RUNG 0003
REPLACE                   %I0007  :              ::
```

If the cut buffer becomes full, repeat the operation, selecting a smaller amount of logic.

3. All rungs below the selected section will scroll up to fill the gap of the just-cut section.

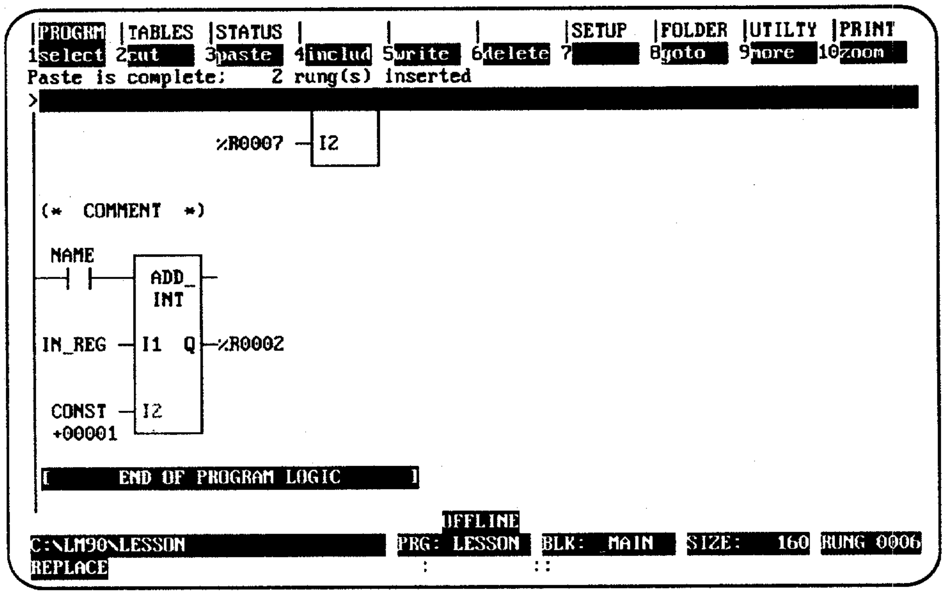
Note

Cut rungs are saved only as long as you remain in the program editor.

Pasting Previously Cut Rungs

The paste function enables you to insert a copy of a section of previously cut rungs before the current rung. Any unique reference address nickname assignments and any identifier name will be added to the current program's variable declaration table.

Once the cut operation is complete and the logic has been placed in the buffer, press **Paste** (**F3**) to initiate the paste function. The position of the cursor remains the same. All entries below the inserted section will scroll down to make room for the just-pasted rungs.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1select 2cut    3paste  4includ 5write  6delete 7       8goto   9more   10zoom
Paste is complete;    2 rung(s) inserted
>

              %R0007 -| I2

 (*  COMMENT  *)

  NAME
 --| |--- ADD_ -
          INT

 IN_REG -| I1  Q|-%R0002

  CONST -| I2
  +00001

 [       END OF PROGRAM LOGIC          ]

                                              OFFLINE
C:\LM90\LESSON                               PRG: LESSON  BLK:  MAIN  SIZE:    160 RUNG 0006
REPLACE                                         :           ::
```

Conflicts may occur between declarations in the section of rungs to be pasted (buffer) and declarations in the existing (target) program. A nickname in the buffer that is identical to one in the target program is simply passed over and ignored during the paste operation.

When a declaration in the buffer has the same reference address but a different nickname than a declaration in the target program, the declaration from the buffer is not added to the existing program's declaration table.

A name conflict occurs when a declaration in the buffer has the same name as a declaration in the target program, but a different meaning. For a nickname, the different meaning would be a different reference address.

Name conflicts are resolved by automatically generating a unique system name for the conflicting declaration in the buffer, and then adding it to the target program. The presence of system names (any name beginning with the **$** character, e.g., $LA00001) in the pasted logic indicates that name collisions have occurred during the paste operation.

Implicit declarations, such as JUMPs, LABELs, and MCR names, are handled in the same way.

Note

If coil checking is set to SINGLE, rung(s) may not be pasted should a coil-use conflict occur. %T references should be used on coils to allow rungs to be pasted, or use the WARN MULTIPLE coil check option. (Refer to the information on coil checking later in this section.) %T references may later be replaced with %Q or %M references.

Writing Selected Rungs to a File

The file write function enables you to store a copy of a section of previously selected rungs from the program logic to a disk file called a side file or program segment. This function is useful for creating a file of commonly used rungs which can be used in different programs.

Only variable declarations used in the rungs that are selected are written to the file. To write the entire variable declaration table to a file, you should select the entire table from the variable declaration section of the program.

Note

The file write function differs from the cut function in that selected rungs are not removed from the program.

1. To use the file write function, you must first select the rungs to be written to the side file. Press **F1** to select the current rung; then, select additional rungs required.
2. Enter the name of the side file on the command line, and press **Write** (**F5**). The selected section of rungs is written out to the specified file.

Any valid file name, minus an extension, can be used for the side file. If the file specification includes a path, the specified directory must already exist. It is not created as part of a file write operation. If no path is specified, the current folder is assumed.

Including Rungs from a File

The file include function enables you to insert a copy of a file of previously written rungs before the current rung in the program logic. All rungs below the inserted section will be scrolled down to make room for the just-included rungs.

Note

Side files which contain subroutines cannot be inserted into folders using Logicmaster 90-30/20/Micro software releases prior to Release 3.

Any unique reference address nickname assignments and any identifier names are added to the current program's variable declaration table. If the program name in the include file declaration matches a program name in the target folder, the duplicate entry is not added. Conflicts are handled as previously described under "Pasting Previously Cut Rungs."

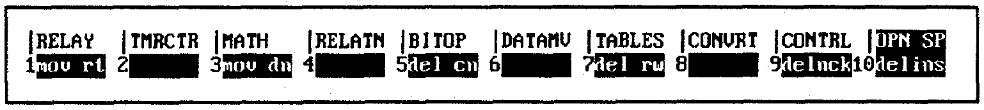
If the program name from the include file conflicts with a non-program name in the target folder, a unique system name is automatically generated for the program declaration before it can be added to the target folder. For example, an MCR has the name END_OP in the include file, but END_OP is also the name of the folder (or a nickname). A new name is generated for END-OP ($MC0001). A corresponding initial logic block and data block are also created.

1. To use the file include function, the program logic to be included must have been previously stored to a particular disk file.
2. Press **`Include`** (**`F4`**) and enter the name of the side file on the command line. Then, press the **`Enter`** key. The contents of the specified disk file are included before the current rung in the program.

Any valid file name, minus an extension, can be used for the side file. The file specification may include a path; however, if no path is specified, the current folder is assumed.

Open Space Functions

Open space functions are used, while editing a rung, to open element spaces in the rung. To display the open space function keys shown below, press **F10**.



```
|RELAY  |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1mov rt 2       3mov dn 4       5del cn 6       7del rw 8       9delnck10delins
```

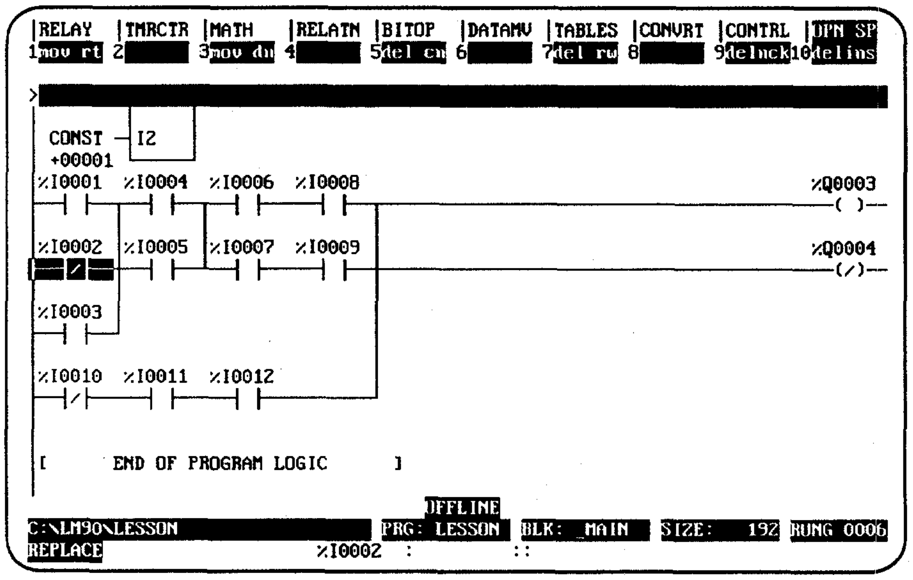
Function Key	Function	Description	Page
F1	Move Right	Move the element at the cursor position, and all the elements to the right of the cursor position, one position to the right in all lines. Shunts are automatically inserted into the new column in the rung.	3-85
F3	Move Down	Create room for an additional line of logic above the line the cursor is on.	3-86
F5	Delete Column	Delete an entire column of instructions (including verticals) and operands within a rung. All elements to the right of the deleted column will automatically move left.	3-87
F7	Delete Row	Delete an entire row of instructions (including verticals) and operands within a rung.	3-88
F9	Delete Nickname	Remove nicknames from the variable declarations table. Enter the nickname to be deleted on the command line before pressing **F9**.	N/A
F10	Delete Instruction	Delete an operand or instruction from a rung. (You may also press **ALT-D** to delete an operand or instruction.)	3-89

Move Logic Right

The Move Right (MOV RT) function moves the element at the cursor position, and all the elements to the right of the cursor position, one position to the right in all lines. Shunts are automatically inserted into the new column in the rung.

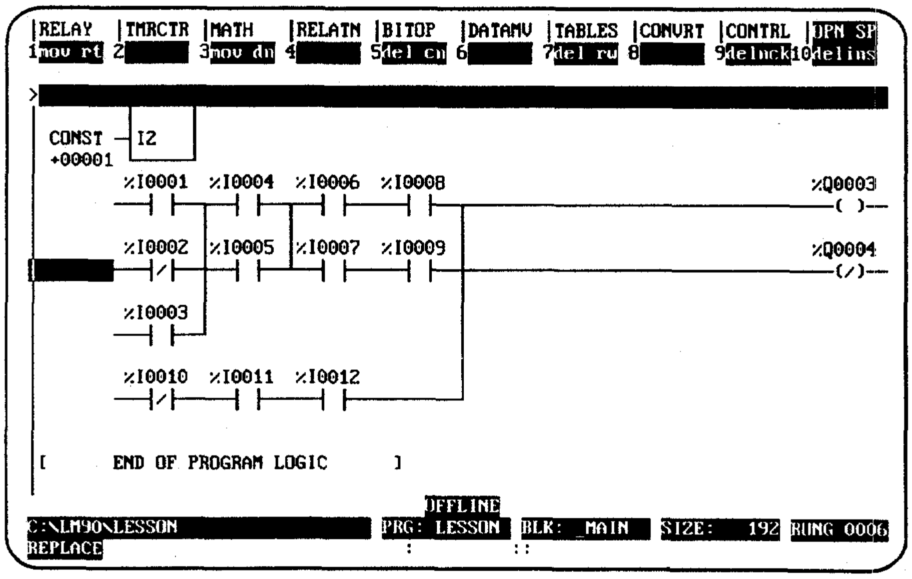
In the following example, the column next to the power rail is moved right one column by positioning the cursor in column 1 and pressing **Move Right** (**F1**).

Before:



```
|RELAY  |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1mov rt 2       3mov dn 4       5del cm 6       7del rw 8       9delnck10del ins
>
 CONST -| I2
 +00001
%I0001  %I0004  %I0006  %I0008                                   %Q0003
                                                                  -( )-
%I0002  %I0005  %I0007  %I0009                                   %Q0004
                                                                  -(/)-
%I0003

%I0010  %I0011  %I0012

[      END OF PROGRAM LOGIC      ]
                                          OFFLINE
C:\LM90\LESSON                            PRG: LESSON  BLK: _MAIN  SIZE:  192  RUNG 0006
REPLACE                            %I0002  :          ::
```

After:



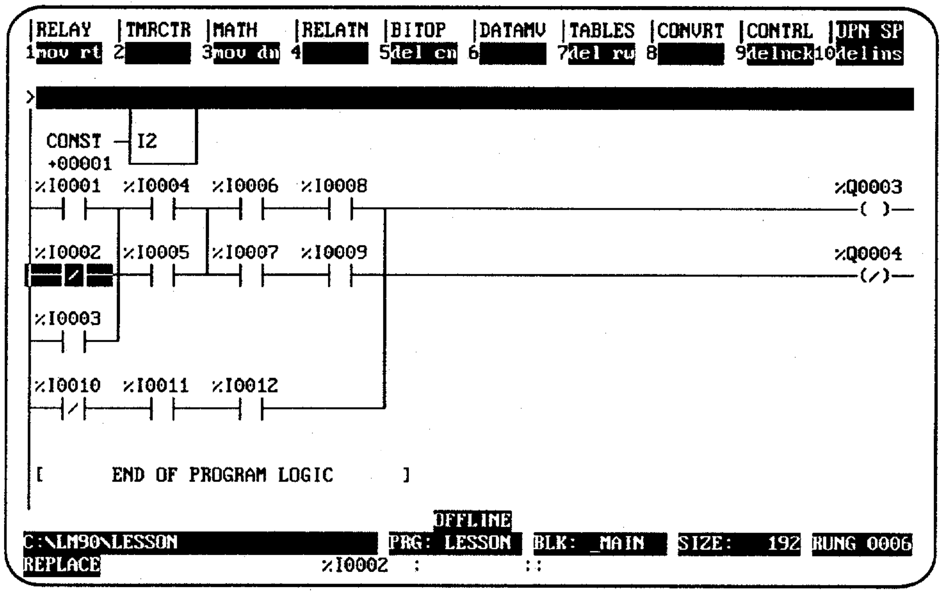
```
|RELAY  |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1mov rt 2       3mov dn 4       5del cm 6       7del rw 8       9delnck10del ins
>
 CONST -| I2
 +00001
        %I0001  %I0004  %I0006  %I0008                           %Q0003
                                                                  -( )-
        %I0002  %I0005  %I0007  %I0009                           %Q0004
                                                                  -(/)-
        %I0003

        %I0010  %I0011  %I0012

[      END OF PROGRAM LOGIC      ]
                                          OFFLINE
C:\LM90\LESSON                            PRG: LESSON  BLK: _MAIN  SIZE:  192  RUNG 0006
REPLACE                                    :          ::
```

Move Logic Down

The Move Down (MOV DN) function is used to create room for an additional line of logic above the line the cursor is on. All elements in the region to be moved must be located entirely in the same row as the cursor, or in the rows below.

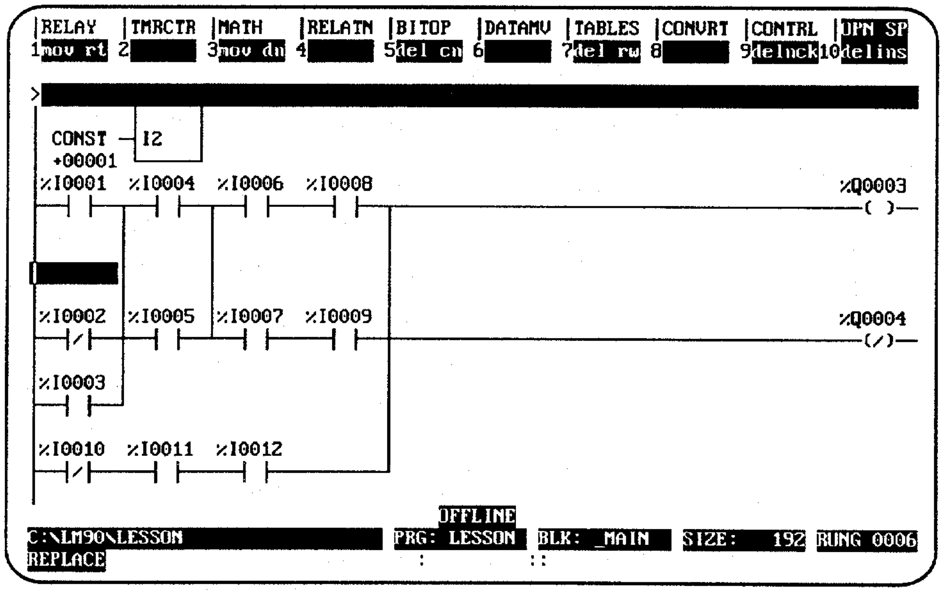
In the following example, the rung of logic beginning with %I0002 is moved down one row by positioning the cursor in row 2 and pressing **Move Down (F3)**.

Before:



|RELAY |TMRCTR |MATH |RELATN |BITOP |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1mov rt 2 3mov dn 4 5del cn 6 7del rw 8 9delnck10del ins
CONST
+00001
I2
%I0001 %I0004 %I0006 %I0008 %Q0003
%I0002 %I0005 %I0007 %I0009 %Q0004
%I0003
%I0010 %I0011 %I0012
[END OF PROGRAM LOGIC]
OFFLINE
C:\LM90\LESSON PRG: LESSON BLK: _MAIN SIZE: 192 RUNG 0006
REPLACE %I0002 : ::

After:



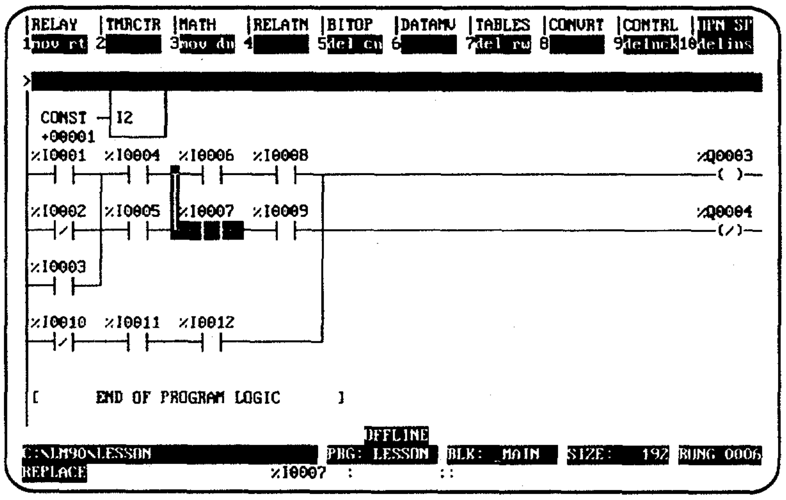
|RELAY |TMRCTR |MATH |RELATN |BITOP |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1mov rt 2 3mov dn 4 5del cn 6 7del rw 8 9delnck10del ins
CONST
+00001
I2
%I0001 %I0004 %I0006 %I0008 %Q0003
%I0002 %I0005 %I0007 %I0009 %Q0004
%I0003
%I0010 %I0011 %I0012
OFFLINE
C:\LM90\LESSON PRG: LESSON BLK: _MAIN SIZE: 192 RUNG 0006
REPLACE : ::

Delete Column

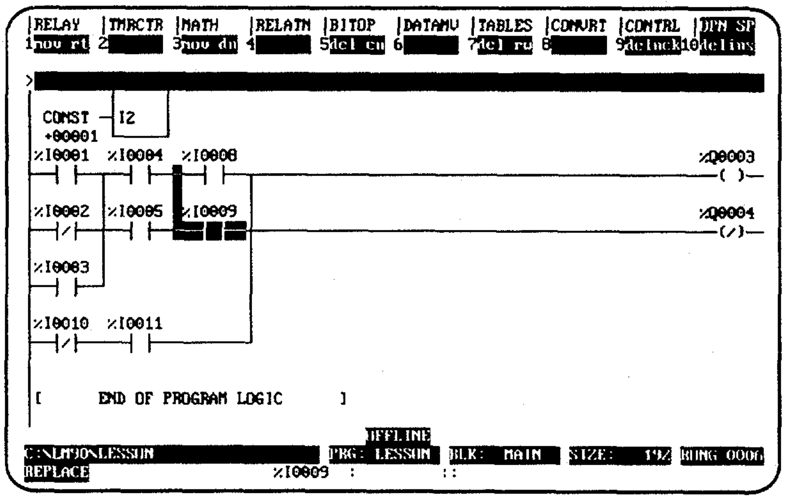
The Delete Column (DEL CN) function is used to delete an entire column of instructions (including verticals) and operands within a rung. It can be used to delete a column that contains functions as well as contacts or coils, as long as the function template itself is totally contained within the column.

When the column to be deleted is within a rung that contains no coil or jump instructions, all columns to the right of the deleted column are moved left one column. In the following example, the column which contains %I0007 is deleted by positioning the cursor in column 3 and pressing **Delete Column** (**F5**).

Before:



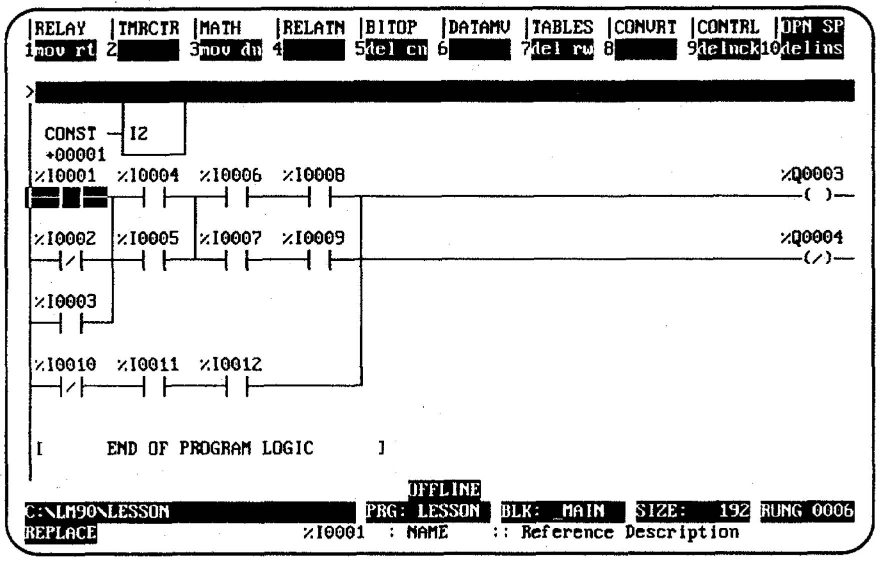
After:



Delete Row

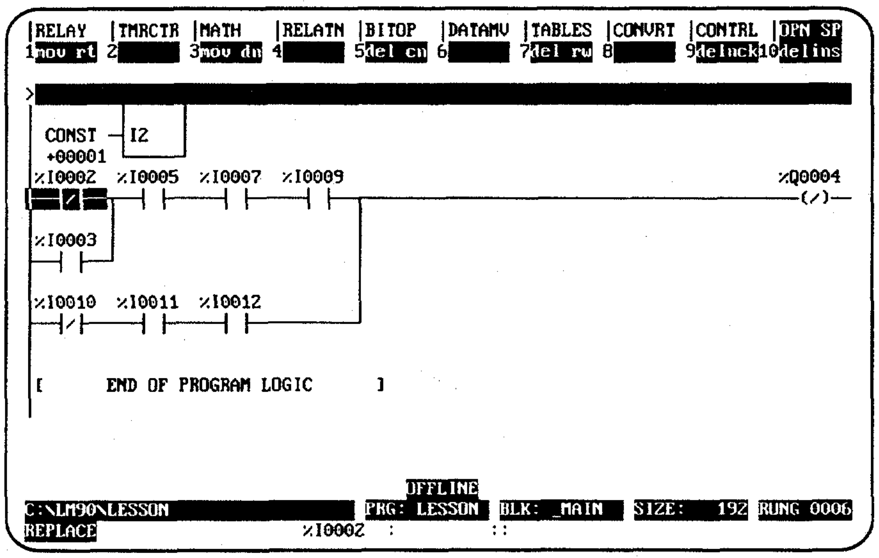
The Delete Row (DEL RW) function is used to delete an entire row of instructions (including verticals) and operands within a rung. It can only be used to delete a row whose instructions, together with their operands, are totally contained within the row. For example, the DEL RW function can be used to delete a row with a function such as MCR, but it cannot delete a row containing an ADD_INT function because the ADD_INT function spans several rows. After the deletion, all rows beneath the deleted row move up one row. When a row is moved up to the first row position, its verticals are automatically deleted. In the following example, the row which contains %I0001 is deleted by positioning the cursor on that row and pressing **Delete Row** (**F7**).

Before:



```
|RELAY  |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1mov rt 2       3mov dn 4       5del cn 6       7del rw 8       9delnck10del ins

 CONST -| I2
 +00001
%I0001  %I0004  %I0006  %I0008                                       %Q0003
%I0002  %I0005  %I0007  %I0009                                       %Q0004
%I0003
%I0010  %I0011  %I0012

[      END OF PROGRAM LOGIC      ]
                                         OFFLINE
C:\LM90\LESSON                          PRG: LESSON  BLK: _MAIN  SIZE:   192 RUNG 0006
REPLACE                       %I0001  : NAME     :: Reference Description
```

After:



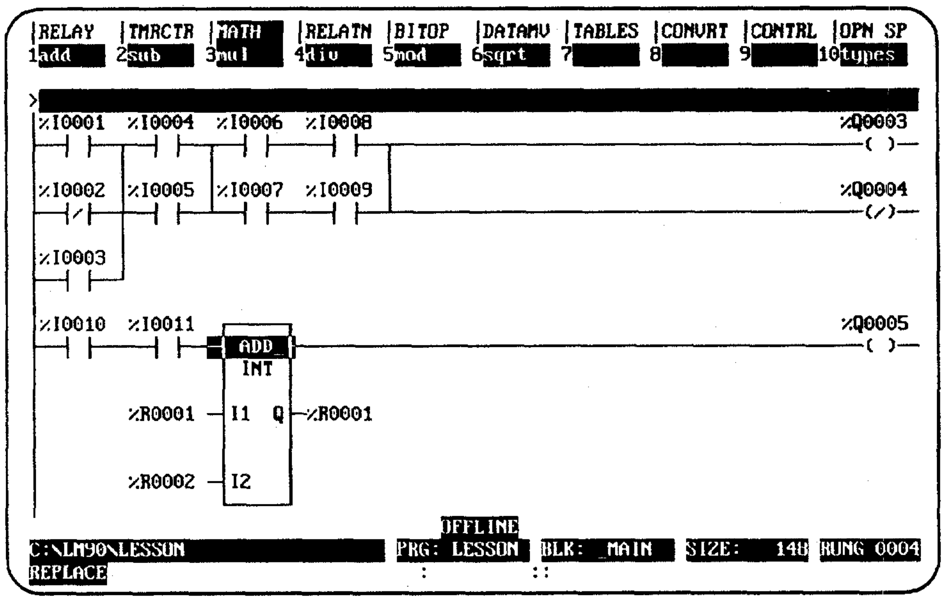
```
|RELAY  |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1mov rt 2       3mov dn 4       5del cn 6       7del rw 8       9delnck10del ins

 CONST -| I2
 +00001
%I0002  %I0005  %I0007  %I0009                                       %Q0004
%I0003
%I0010  %I0011  %I0012

[      END OF PROGRAM LOGIC      ]
                                         OFFLINE
C:\LM90\LESSON                          PRG: LESSON  BLK: _MAIN  SIZE:   192 RUNG 0006
REPLACE                       %I0002  :          ::
```

Delete Instruction

The Delete Instruction (DELINS) function is used to delete an operand or instruction from a rung. Another instruction or a horizontal link may then be entered in the space left by the deleted instruction. The ALT-D key sequence may also be used to delete instructions; however, the cursor will automatically move to the right after the deletion.

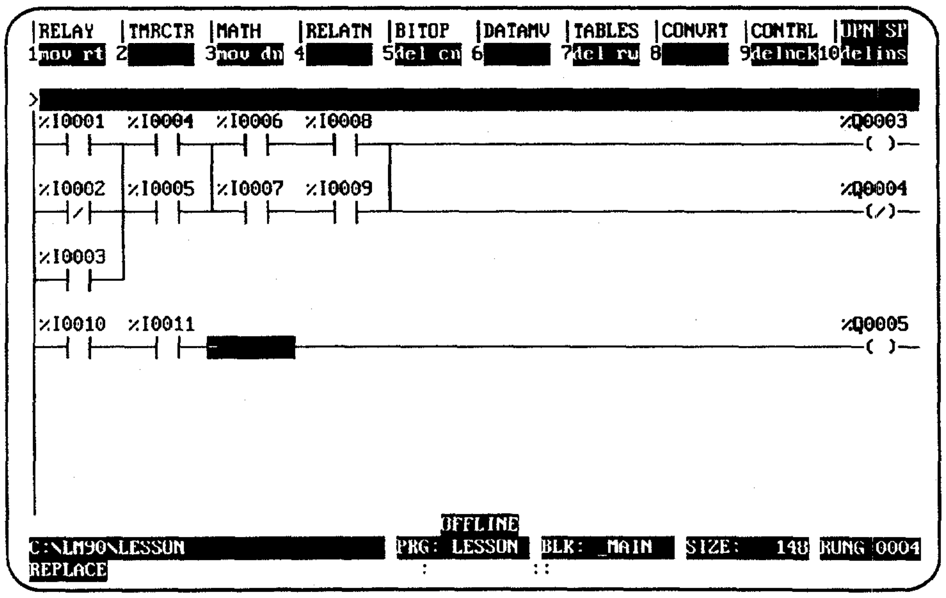
In the following example, the ADD instruction is deleted by positioning the cursor on the ADD function block and pressing **Delete Instruction** (**F10**).

Before:



|RELAY |TMRCTR |MATH |RELATN |BITOP |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1add 2sub 3mul 4div 5mod 6sqrt 7 8 9 10types
%I0001 %I0004 %I0006 %I0008 %Q0003
%I0002 %I0005 %I0007 %I0009 %Q0004
%I0003
%I0010 %I0011 %Q0005
ADD_
INT
%R0001 I1 Q %R0001
%R0002 I2
OFFLINE
C:\LM90\LESSON PRG: LESSON BLK: _MAIN SIZE: 148 RUNG 0004
REPLACE

After:



|RELAY |TMRCTR |MATH |RELATN |BITOP |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1mov rt 2 3mov dn 4 5del cn 6 7del ru 8 9delnck 10delins
%I0001 %I0004 %I0006 %I0008 %Q0003
%I0002 %I0005 %I0007 %I0009 %Q0004
%I0003
%I0010 %I0011 %Q0005
OFFLINE
C:\LM90\LESSON PRG: LESSON BLK: _MAIN SIZE: 148 RUNG 0004
REPLACE

Increment/Decrement Reference Address

The increment/decrement reference address feature is available in either **INSERT** or **EDIT** mode in the program editor and from the variable declaration editor. It allows you to increment or decrement the reference address where the cursor is located by either one or the amount specified on the command line.

Key	Description
CTRL-U	Increment key sequence.
CTRL-D	Decrement key sequence.

Note

For reference operands which must be byte-aligned, the reference address is incremented/decremented by one byte.

For example, a discrete input reference address of a MOVE_INT function block will increment or decrement by one byte (e.g., %I0001 to %I0009). The same reference address on a normally open contact will increment or decrement by one (e.g., %I0001 to %I0002). A register reference address on the input of an ADD function block will also increment or decrement by one (e.g., %R0005 to %R0006).

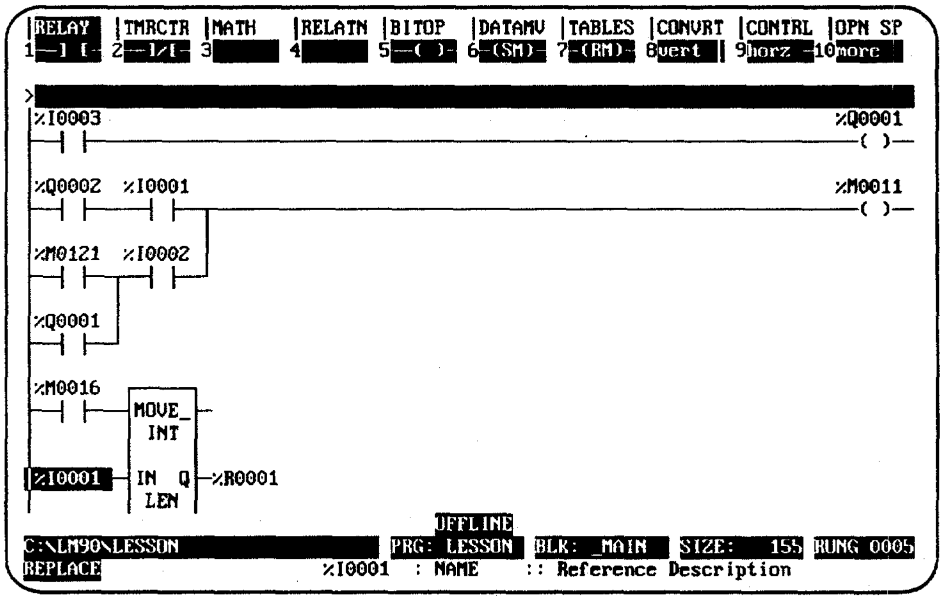
If there is a numeric value on the command line, it is used as the number of bits to add to or subtract from the reference address. For parameters that are byte-aligned, the increment/decrement value is one byte when the value on the command line is between 0 and 8, two bytes when the value is between 9 and 16, etc. The value will remain displayed on the command line after the function is completed so that the reference address can be incremented/decremented by the same value again.

If the reference address reaches the current configuration limit, an error message is displayed and the function is not performed. If the cursor is not on a reference address and you attempt to increment or decrement the address, an error message is also displayed.

Incrementing/Decrementing within a Rung

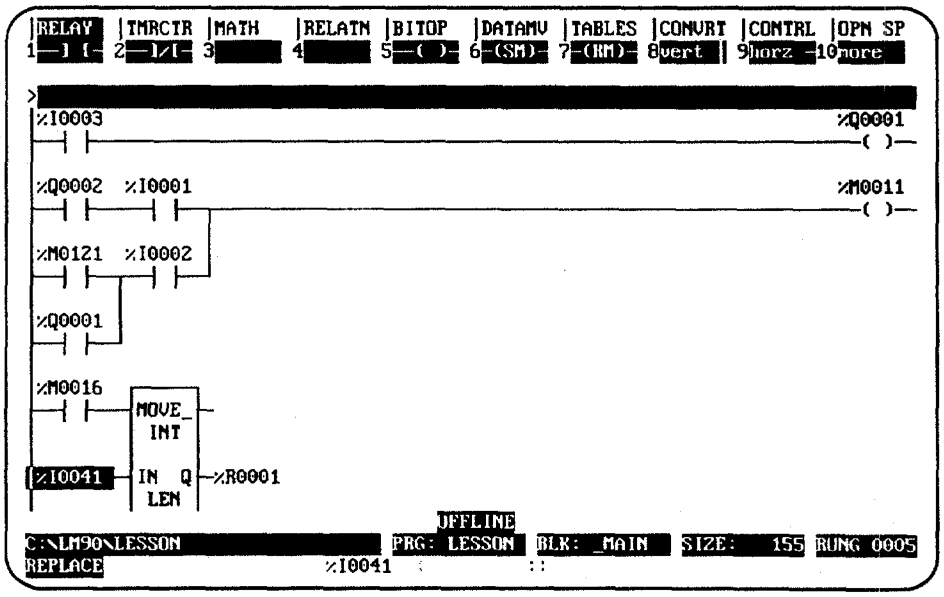
This example illustrates how to increment reference address %I0001 in a ladder diagram rung.

1. First, position the cursor on the reference address to be changed. In this example, that would be %I0001.



RELAY | TMRCTR | MATH | RELATN | BITOP | DATAMV | TABLES | CONVRT | CONTRL | OPN SP
1 -] [- 2 -]/[- 3 4 5 -( )- 6 -(SM)- 7 -(RM)- 8 vert 9 horz 10 more
%I0003
%Q0001
%Q0002
%I0001
%M0011
%M0121
%I0002
%Q0001
%M0016
MOVE_ INT
%I0001
IN Q
%R0001
LEN
OFFLINE
C:\LM90\LESSON
PRG: LESSON
BLK: _MAIN
SIZE: 155
RUNG 0005
REPLACE
%I0001 : NAME :: Reference Description

2. Press **CTRL-U** five times to display the reference address %I0041, as shown in this screen.



3. To decrement the reference address to %I0025, press **CTRL-D** twice. Or, you could enter the decrement amount 16 (2 bytes) on the command line and press **CTRL-D**.

   The value 16 displayed on the command line does not disappear when the decrement is completed. It continues to be displayed on the command line so that you may continue decrementing by that same amount.

Auto-Next Highest Reference

The auto-next highest reference address function automatically uses the next reference offset of the specified type, after the highest currently used in the folder during an editing session. If the parameter requires a byte-aligned reference offset, the next available aligned offset is automatically provided.

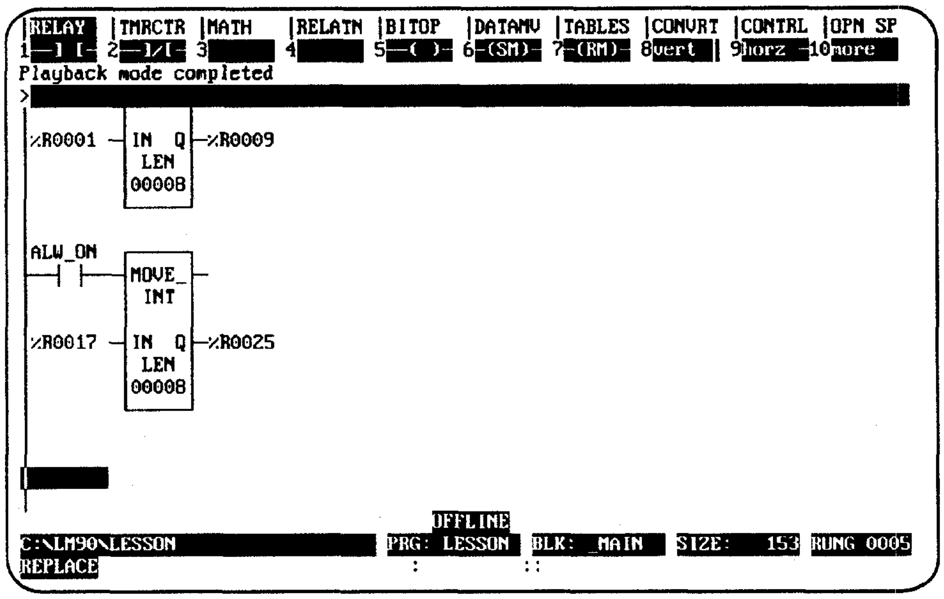
To use this function, enter the % character and the user reference (e.g., %I, %Q, %R, etc.) on the command line. The % character is used to distinguish a reference type from a nickname character. Then, press the **Enter** key.

For example, if the highest %I reference already used during an editing session is %I0019 and %I is specified as the address for an input operand to an ADD_INT function block, %I0024 (the next available aligned reference) is automatically used.

The following example illustrates how useful the auto-next highest reference address function can be, when used in conjunction with **TEACH** mode. It allows rungs to be entered and reference addresses automatically assigned to the next available reference address. (For more information on **TEACH** mode, refer to chapter 2, section 5, "Keyboard Functions").

1. Enter **INSERT** mode by pressing **Insert** (**F1**) from the program editor.
2. Press **ALT-T** to enter **TEACH** mode, and then press **ALT-N**, where **n** is the number to be used for the teach file. For this example, use **ALT-0**.
3. On the command line, type **ALW_ON &NOCON** and press the **Enter** key.
4. Then, type **&MOV** on the command line and press the **Enter** key.
5. Use the **Tab** key to position the cursor on parameter IN of the MOVE function block.
6. Type **%R** on the command line, and press the **Tab** key.
7. Then, enter **%R** on the command line again (or press **CTRL-Home** to display the previous command line contents), and press the **Enter** key twice.
8. Press **ALT-Q** to exit **TEACH** mode.

9. Press **ALT-O** to play back the teach file. The following screen is displayed. Note that the reference address has automatically been changed to the next available address.



```
|RELAY  |TMRCTR |MATH    |RELATN |BITOP   |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1-] [- 2-]/[- 3        4       5-( )-  6-(SM)- 7-(RM)- 8vert |  9horz -10more
Playback mode completed
>

%R0001 -|IN  Q|-%R0009
        | LEN |
        |00008|

ALW_ON
--| |---|MOVE_|-
        | INT |

%R0017 -|IN  Q|-%R0025
        | LEN |
        |00008|

                                    OFFLINE
C:\LM90\LESSON                  PRG: LESSON  BLK: _MAIN  SIZE:   153 RUNG 0005
REPLACE
```

This page is intentionally left blank.

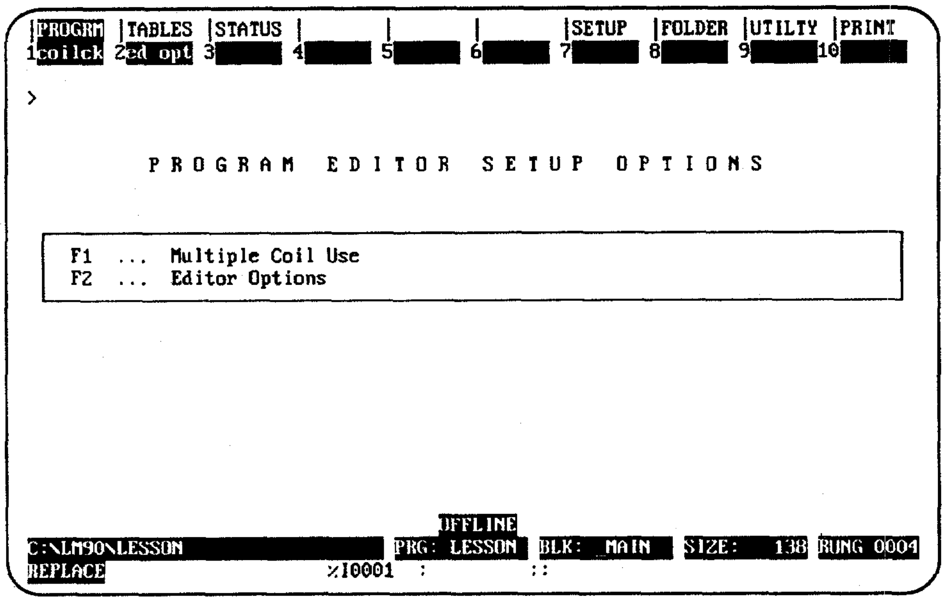
Section 10: Editor Options

The Program Editor Options menu provides access to options in the program editor. These options include multiple coil use and automatically inserting references.

Coil Checking

The coil check function of the programming software checks for multiple uses of %M or %Q references with relay coils or outputs on functions. Beginning with Release 3 of Logicmaster 90-30/20/Micro software, you can select the level of coil checking desired from a screen similar to the one shown below.

To access this screen, press **Program** (**F1**) from the Programming Software main menu. Then, press **Options** (**F7**) from the program editor. Instructions for selecting the desired level of coil checking begin on page 3-96.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1coilck 2ed opt 3       4       5       6       7       8       9       10
>

        P R O G R A M   E D I T O R   S E T U P   O P T I O N S

  F1  ...  Multiple Coil Use
  F2  ...  Editor Options

                                              OFFLINE
C:\LM90\LESSON                           PRG: LESSON  BLK:  MAIN  SIZE:   138  RUNG 0004
REPLACE                    %I0001  :               ::
```

Three levels of coil checking are available:

Level	Name	Description
SINGLE	Coil checking enabled	Single coil use only. When coil use is set to **SINGLE**, Logicmaster 90-30/20/Micro software does not allow multiple coil use. When the rung is accepted and multiple use is detected, an error message is displayed and the cursor is placed on the first conflict found. You cannot exit the rung edit until the conflicts are resolved. When this setting is accepted, the coil use map is rebuilt. A "checking coil use" message is displayed while the map is rebuilt. If conflicts are found, only the reference address of the conflict is listed. Only one screen full of conflicts is displayed. If conflicts are found, the ***Current Coil Use*** field is not updated; it is only updated when no conflicts are found.
WARN MULTIPLE*	Coil checking disabled with warning	When coil use is set to **WARN MULTIPLE**, Logicmaster 90-30/20/Micro software allows multiple coil use with warning messages. The ***Current Coil Use*** field is updated, and the coil use map is rebuilt. A message is displayed while the map is rebuilt and the conflicts are listed. Only one screen of conflicts is displayed.
MULTIPLE	Coil checking disabled	When coil use is set to **MULTIPLE**, Logicmaster 90-30/20/Micro software allows multiple coil use without any restrictions or messages. The ***Current Coil Use*** field is updated, and the coil use map is not rebuilt. No conflicts are displayed.

* Default selection.

Note

If the program folder is locked, the level of coil checking cannot be changed and the coil map cannot be rebuilt. Refer to the information on unlocking program folders described in chapter 7, "Program Folders."

The Coil Check screen shows the current level of coil checking, allows you to change the level, and checks for and displays coil use conflicts on demand. You can change the level selected, regardless of which level was previously selected.

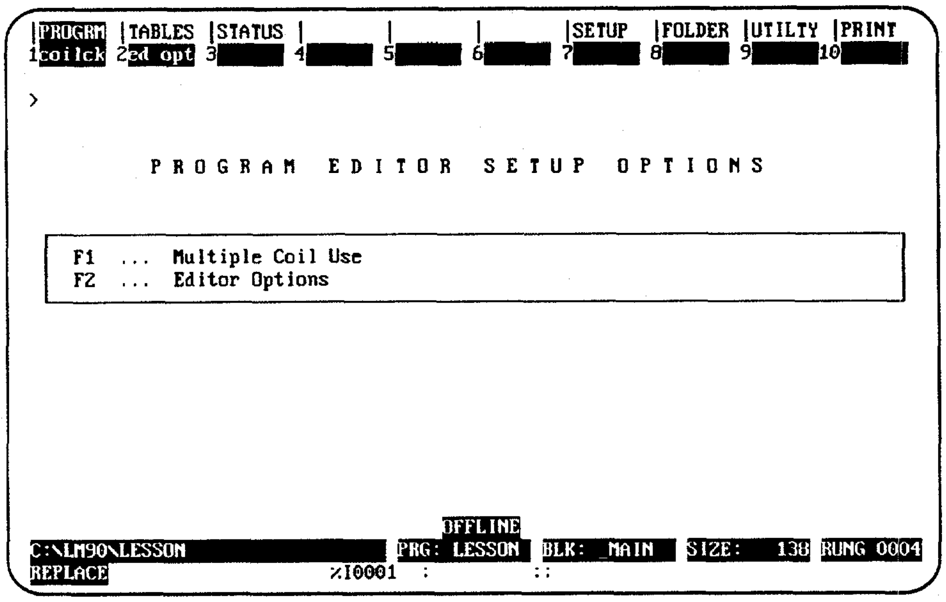
Coils can function as SET Coils or as RESET Coils with MULTIPLE or WARN MULTIPLE coil checking enabled. For details on the possible effects of MULTIPLE and WARN MULTIPLE checking on Coils, refer to the information on "SET Coils" and "RESET Coils" in section 1, "Relay Functions," of chapter 3 of the *Series 90™-30/20/Micro Reference Manual* (GFK-0467).

Note

When the program mode is **MONITOR** or **ONLINE** and the program in your computer is identical to the program in the PLC (i.e., **LOGIC EQUAL** status is displayed), only the current level of coil checking is displayed. An error message is displayed if you try to change the coil check level.

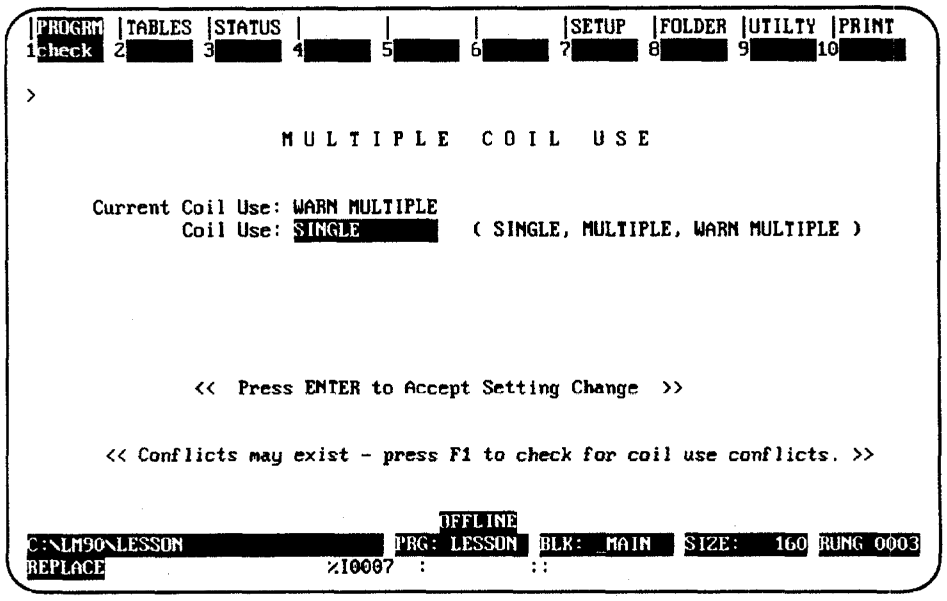
To change the level and check for conflicts:

1. Press **Program** (**F1**) from the Programming Software main menu. Then, press **Options** (**F7**) from the program editor to display the Program Editor Options menu.



```
|PROGRM |TABLES |STATUS |      |      |      |SETUP  |FOLDER |UTILTY |PRINT
1coilck 2ed opt 3       4      5      6      7       8       9       10
>

            P R O G R A M   E D I T O R   S E T U P   O P T I O N S

   F1  ...  Multiple Coil Use
   F2  ...  Editor Options

                                              OFFLINE
C:\LM90\LESSON                               PRG: LESSON  BLK: MAIN  SIZE:   138 RUNG 0004
REPLACE                          %I0001  :          ::
```

2. Press **Coil Check** (**F1**) to display the Multiple Coil Use screen.



```
|PROGRM |TABLES |STATUS |      |      |      |SETUP  |FOLDER |UTILTY |PRINT
1check  2       3       4      5      6      7       8       9       10
>
                     M U L T I P L E   C O I L   U S E

     Current Coil Use: WARN MULTIPLE
             Coil Use: SINGLE          ( SINGLE, MULTIPLE, WARN MULTIPLE )



             <<  Press ENTER to Accept Setting Change  >>

     << Conflicts may exist - press F1 to check for coil use conflicts. >>

                                              OFFLINE
C:\LM90\LESSON                               PRG: LESSON  BLK: MAIN  SIZE:   160 RUNG 0003
REPLACE                          %I0007  :          ::
```

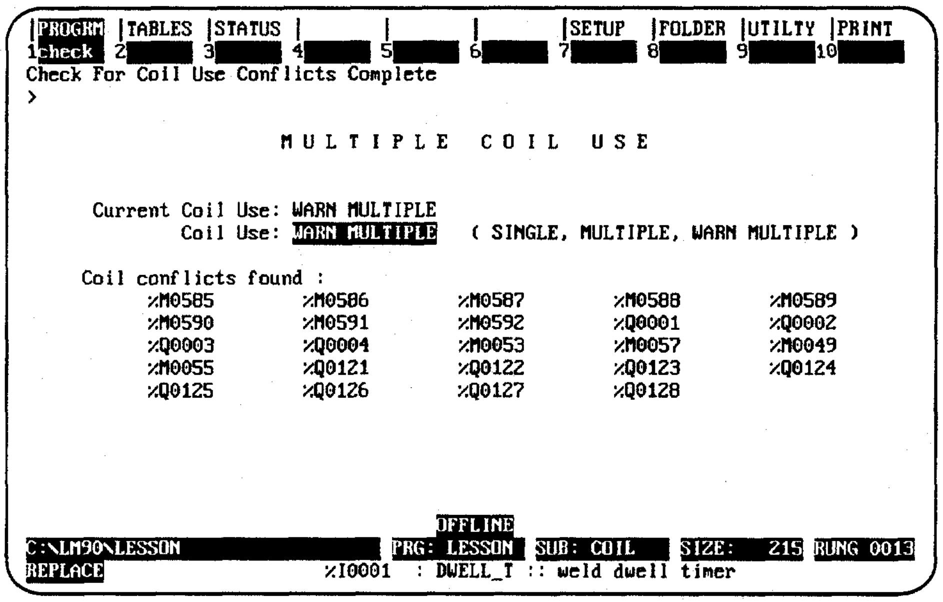
3. The screen shows the current level of coil checking. To select another level, repeatedly press the **Tab** key until the desired setting is displayed. Then, press the Enter key to accept the change. Or, you can type the desired setting into the *Coil Use* field, and then press the **Enter** key.

4. The coil use map is automatically rebuilt when the **SINGLE** or **WARN MULTIPLE** level is accepted. To rebuild the coil map without changing the current setting, press **Check** (**F1**). *F1* may be used regardless of the current coil use setting. The message "Checking Block . . . " is displayed as the map is built, along with the name of the block currently being checked.

   The rebuilding of the map takes approximately 15 seconds per block. For example, a program consisting of 10 to 15 blocks may take up to two minutes to rebuild the map.

5. Press **ALT-A** to abort the rebuild and restore the old coil use map. A message indicating that an abort has occurred is displayed.

6. When the rebuild is complete, any conflicts found are listed. Both explicit and implicit use conflicts are displayed. Only one screen full of conflicts is displayed.


|PROGRM |TABLES |STATUS | | | |SETUP |FOLDER |UTILTY |PRINT
1check 2 3 4 5 6 7 8 9 10

```
Check For Coil Use Conflicts Complete
>
                    M U L T I P L E   C O I L   U S E

     Current Coil Use: WARN MULTIPLE
             Coil Use: WARN MULTIPLE   ( SINGLE, MULTIPLE, WARN MULTIPLE )

    Coil conflicts found :
          %M0585      %M0586      %M0587      %M0588      %M0589
          %M0590      %M0591      %M0592      %Q0001      %Q0002
          %Q0003      %Q0004      %M0053      %M0057      %M0049
          %M0055      %Q0121      %Q0122      %Q0123      %Q0124
          %Q0125      %Q0126      %Q0127      %Q0128

                                       OFFLINE
C:\LM90\LESSON                     PRG: LESSON  SUB: COIL   SIZE:   215 RUNG 0013
REPLACE                  %I0001  : DWELL_T :: weld dwell timer
```

Conflicts are not saved in memory. When you leave this screen, they will not be redisplayed if you re-enter this display screen. However, the **Check** (**F1**) softkey may be pressed to check for coil conflicts.

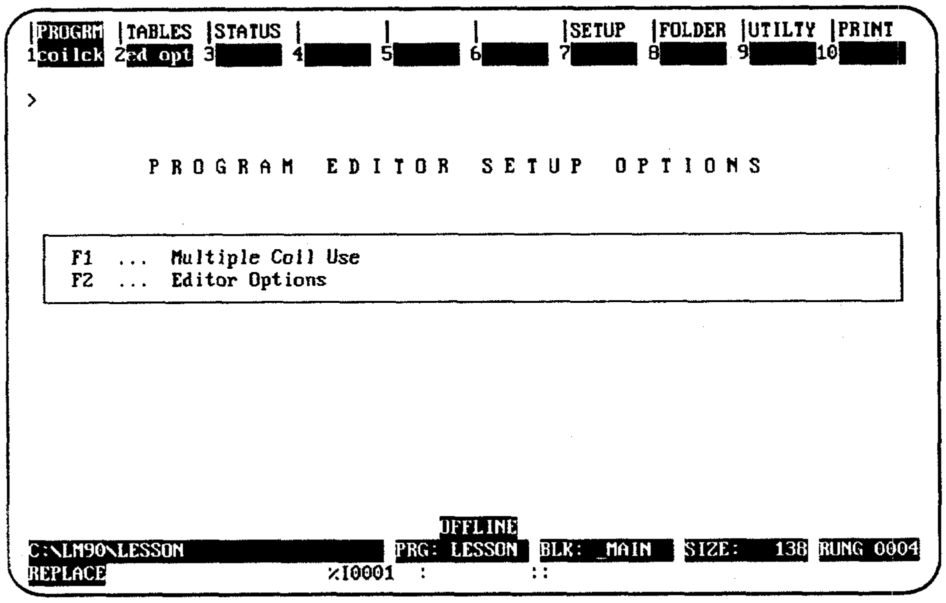
ALT-P may be used to print a screen of conflicts. For information on printing a screen display, refer to chapter 9, section 2, "Selecting a Screen Print Device."

Automatically Inserting References

References which do not have nicknames or reference descriptions can be automatically inserted into the variable declaration table as the program is being developed. Then, at a later time, you can go to the variable declaration editor and enter just the annotation.

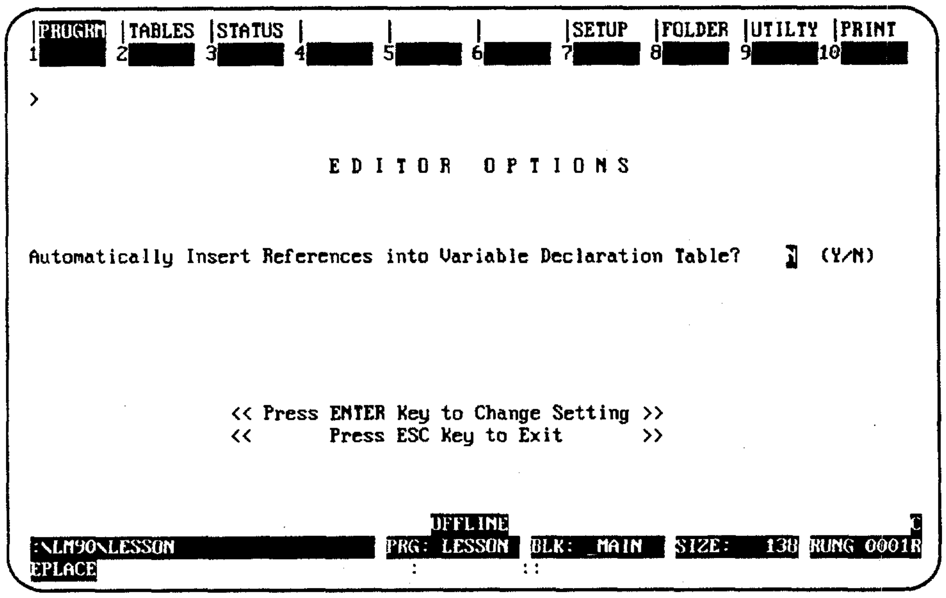
To enable the automatic insertion of references:

1. From the Programming Software main menu, press **Program** (**F1**) and then **Options** (**F7**) to display the Program Editor Options menu.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1coilck 2ed opt 3       4       5       6       7       8       9       10
>

        P R O G R A M   E D I T O R   S E T U P   O P T I O N S

  F1  ...  Multiple Coil Use
  F2  ...  Editor Options

                                               OFFLINE
C:\LM90\LESSON                              PRG: LESSON  BLK: _MAIN  SIZE:    138 RUNG 0004
REPLACE                        %I0001  :            ::
```

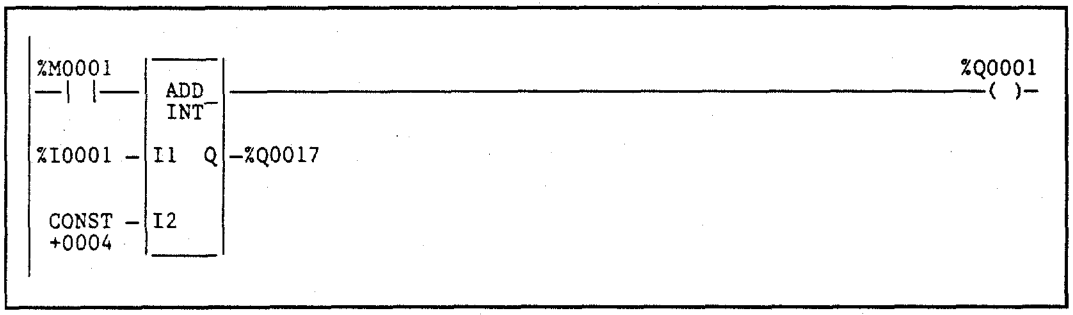
2. Press **Editor Options** (**F2**) to display the Editor Options screen.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1       2       3       4       5       6       7       8       9       10
>

                          E D I T O R   O P T I O N S

Automatically Insert References into Variable Declaration Table?   N  (Y/N)

                     << Press ENTER Key to Change Setting >>
                     <<       Press ESC Key to Exit       >>

                                               OFFLINE                          C
:\LM90\LESSON                               PRG: LESSON  BLK: _MAIN  SIZE:    138 RUNG 0001R
EPLACE                                 :            ::
```

3. Enter **Y** (Yes) and press the **Enter** key to enable automatic insertion. Explicit references entered on the command line during an edit session are automatically inserted in the variable declaration table; implicit references (those references not entered while programming logic) are not automatically inserted in the table.

   In the following example, the explicit reference %I0001 is inserted automatically in the variable declaration table. %I0002 through %I0016 are implicit references and are not inserted automatically in the table.



   In addition, %S and %T references are not added to the table.

   References associated with rungs that are pasted or included are added to the variable declaration table only if this operation were enabled when the rungs were cut or written. Removing rungs using either the cut/paste function or the file include function does not affect the table. Any reference to be removed from the variable declaration table must be deleted using the variable declaration table editor. References entered during a word-for-word operation are not automatically inserted into the variable declaration table.

4. If the variable declaration table becomes full, a message is continuously displayed for each reference entered until some of the references are deleted. Use the variable declaration table editor to delete some references in order to make room for more insertions.

5. To disable the automatic insertion feature, enter **N** (No). When this feature is disabled, references entered on the command line are not placed into the variable declaration table, unless they have a nickname or reference description associated with them.

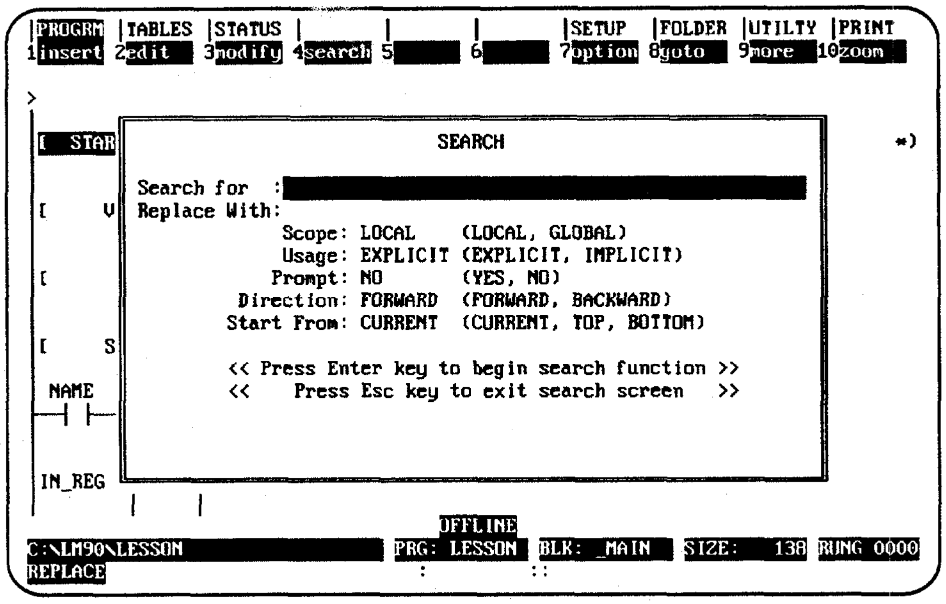
Note

The default selection for automatically inserting references is **N** (No). Every new folder defaults to **NO**.

Section 11: Search Function

The search function enables you to locate an identifier name, reference address, nickname, instruction, instruction plus reference address, instruction plus nickname, instruction plus identifier name, and reference description anywhere in the program.

1. To begin the search function, select **Search** (**F4**) from the program edit functions. When **F4** is pressed, a search function window is displayed on the screen.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3modify 4search 5       6       7option 8goto   9more  10zoom
>
[  STAR                         SEARCH                                    *)

        Search for  :
[     U Replace With:
                  Scope: LOCAL    (LOCAL, GLOBAL)
                  Usage: EXPLICIT (EXPLICIT, IMPLICIT)
[                Prompt: NO       (YES, NO)
              Direction: FORWARD  (FORWARD, BACKWARD)
             Start From: CURRENT  (CURRENT, TOP, BOTTOM)
[     S
             << Press Enter key to begin search function >>
 NAME        <<    Press Esc key to exit search screen    >>

IN_REG
                                       OFFLINE
C:\LM90\LESSON                         PRG: LESSON  BLK: _MAIN   SIZE:    138  RUNG 0000
REPLACE
```

2. To continue searching for the same target, press **ALT-F4**.

   Pressing only the **Search** key accesses the search function again. Since all fields are already set as desired, simply press the **Enter** key to initiate the search.

3. If the target is not found, a message indicating this is displayed. The search function will remain active.

4. If the target is found in a block that is locked for view, the following message is displayed:

   ```
   Found in locked block <block_name>.  (Continue/Quit)
   ```

   If you continue, all remaining blocks are searched. If you decide to quit, the search is aborted. For more information on viewing locked blocks, refer to chapter 3, section 8, "Subroutine Blocks."

5. The search can be aborted at any time by pressing **ALT-A** (abort). A message acknowledging the abort is displayed, and the search function will remain active.

Refer to the definitions in the following table when making entries in the search function window. To change a selection in one of the fields, other than the ***Search for*** and ***Replace with*** fields, use the Tab key to toggle through the available choices.

Field	Description
Search for	The target to be searched for. It may be an identifier name, a reference address (e.g., %I0012) or nickname (e.g., WIDGET), an instruction (e.g., &COIL), an instruction plus reference address (e.g., & COIL %Q0001), nickname (e.g., &COIL WIDGET), or identifier name (e.g., &JUMP BLK1), or a reference description entered in quotes. Enter the desired target into this field. Search allows you to locate all uses of a nickname. First, the software searches the local table if you are in a subroutine block. Next, it searches the main variable declarations table and finally the reserved table. The search operation stops when the first occurrence of the search target is found.
Replace with	What will replace the target being searched for.
Scope	Modify the search by specifying whether the target should be searched for in the current subroutine block only or across all subroutine blocks in the program. Choices for this field include **LOCAL** for the current block only or **GLOBAL** f or all blocks. The search order for all blocks is the subroutine block declaration order, beginning with the current block.
Usage	Modify the search by specifying whether only explicit usage of the reference is checked for (**EXPLICIT**) or both explicit and implicit usage (**IMPLICIT**). More information about implicit search can be found at the end of the search function.
Prompt	Action to be taken if the search target is found. For the search function, selecting **Y** (Yes) indicates that the system will prompt you for confirmation before searching for the next target. When **Y** (Yes) is selected and the target is found, you may display the search target, disregard this instance of the target and continue searching, or terminate the actual search but remain in the search function. For the search and replace function, selecting **Y** (Yes) indicates that the system will prompt you for confirmation before the found target is replaced. There are four choices available: 1. Replace the current found target and continue. 2. Disregard the found target and continue. 3. Starting from this found target, replace the rest of the found targets without prompting (i.e., change the prompt value to **N** (No)). 4. Terminate the replace but remain in the function.
Direction	The direction of the search, either **FORWARD** or **BACKWARD**.
Start From	The starting point of the search, either from the current position (**CURRENT**), from the top of the block (**TOP**), or from the bottom of the block (**BOTTOM**).

The following table shows the legitimate replacement items for the different types of search items.

Search Item	Replacement Item
Reference address or nickname	Reference address or reference nickname.
Identifier	Identifier of the same type.
Instruction	Word-for-word equivalent instruction. Refer to chapter 3, section 12, "Online Editing/Monitoring," for more information on making word-for-word changes.
Instruction and reference	Word-for-word instruction or reference or nickname. Refer to chapter 3, section 12, "Online Editing/Monitoring," for more information on making word-for-word changes.
Instruction and identifier	Identifier of the same type.
Description	No replacement allowed.

Note

If a program contains references that have the same reference description text, searching for the reference description always finds the first occurrence of the reference description.

Using Search and Replace

The search and replace function is used to search and replace the target within the same variable declaration table. It cannot be used to search for the target in one variable table and replace it in another variable table.

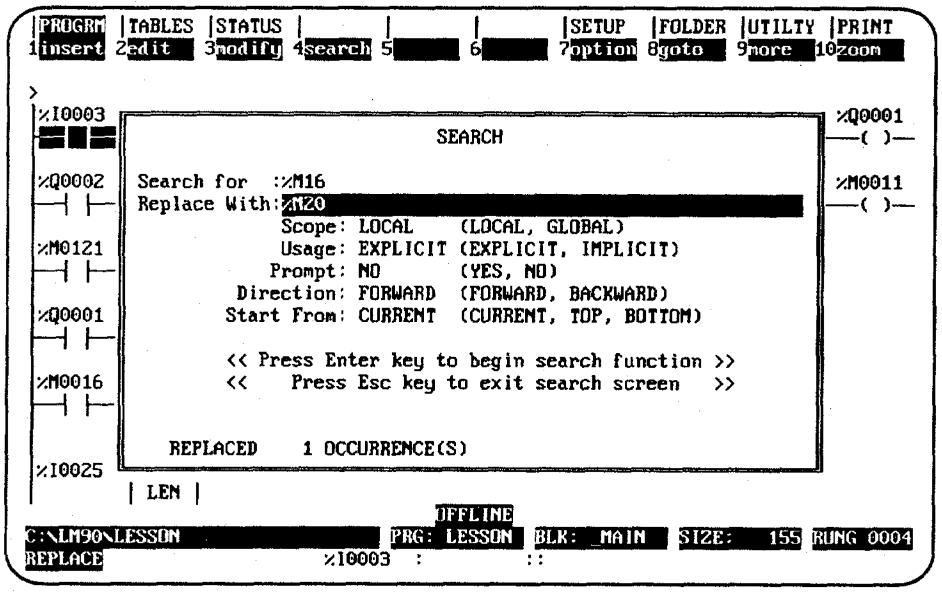
To use the search and replace function:

1. Enter the target to be searched for in the *Search for* field and the replacement item in the *Replace with* field. Other fields can be changed to further modify the search.
2. Press the **`Enter`** key to begin the search function, or press the **`Escape`** key to exit this screen. If the *Prompt* field is set to **`No`**, a "busy" prompt is displayed on the search screen with the current replacement count, indicating that the search and replace function is in progress.
3. If the target is found in a block that is locked for view, the following message is displayed:

```
Cannot write to locked block <block_name>.  (Continue/Quit)
```

You may continue the search and replace operation at the next block or terminate the search and replace. For more information on viewing locked blocks, refer to chapter 3, section 8, "Subroutine Blocks."

In the following example, %M0016 will be searched for and replaced with %M0020. Since the *Prompt* field is set to **`No`**, no confirmation is required. This screen shows the parameters selected for this search.


|PROGRM |TABLES |STATUS | | | |SETUP |FOLDER |UTILTY |PRINT
1insert 2edit 3modify 4search 5 6 7option 8goto 9more 10zoom
%I0003
%Q0002
%M0121
%Q0001
%M0016
%I0025
%Q0001
%M0011
| LEN |

```
                              SEARCH

Search for  :%M16
Replace With:%M20
            Scope: LOCAL     (LOCAL, GLOBAL)
            Usage: EXPLICIT  (EXPLICIT, IMPLICIT)
           Prompt: NO        (YES, NO)
        Direction: FORWARD   (FORWARD, BACKWARD)
       Start From: CURRENT   (CURRENT, TOP, BOTTOM)

          << Press Enter key to begin search function >>
          <<    Press Esc key to exit search screen   >>

   REPLACED     1 OCCURRENCE(S)
```

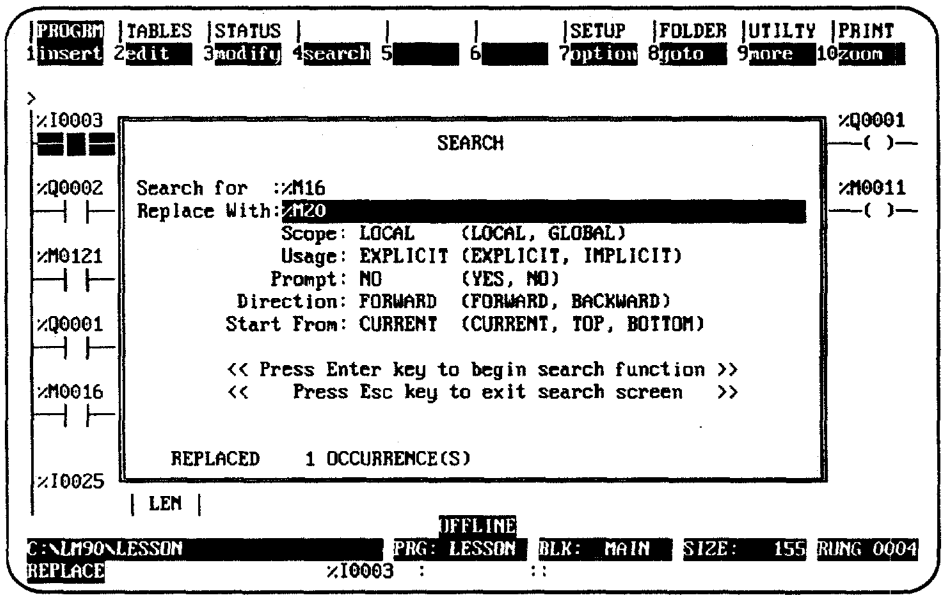
```
                                          OFFLINE
C:\LM90\LESSON                          PRG: LESSON  BLK: _MAIN  SIZE:    155 RUNG 0004
REPLACE                          %I0003  :           ::
```

4. If the *Prompt* field is set to **Y** (Yes), confirmation is required before each replacement. The following prompt is displayed each time the found target (in this example, %M0016) is found:

```
Replace %M0016 with %M0020 ?  (Yes, No, All, Quit)
```

Press **Y** (Yes) to replace the current found target with the replacement item and then continue the search. Select **N** (No) to disregard this instance of the target and continue the search. Press **A** (All) to drop the confirmation process, beginning with this found target, and proceed to replace the rest of the found targets with the replacement item. Pressing **Q** (Quit) terminates the current search and replace process, but remains in the function.

5. Assuming that the function replaced one occurrence of %M0016 with %M0020 without error, an appropriate message including the replacement count is displayed and the function remains active.



```
|PROGRM |TABLES |STATUS |      |      |      |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3modify 4search 5      6      7option 8goto   9more  10zoom
>
|%I0003  ________________________________________________________  %Q0001
|       |                       SEARCH                          |---( )---
|       |                                                       |
|%Q0002 | Search for   :%M16                                    | %M0011
|--| |--| Replace With:%M20                                     |---( )---
|       |           Scope: LOCAL     (LOCAL, GLOBAL)            |
|%M0121 |           Usage: EXPLICIT (EXPLICIT, IMPLICIT)        |
|--| |--|          Prompt: NO        (YES, NO)                  |
|       |       Direction: FORWARD   (FORWARD, BACKWARD)        |
|%Q0001 |      Start From: CURRENT   (CURRENT, TOP, BOTTOM)     |
|--| |--|                                                       |
|       |      << Press Enter key to begin search function >>   |
|%M0016 |      <<    Press Esc key to exit search screen    >>  |
|--| |--|                                                       |
|       |                                                       |
|       |    REPLACED     1 OCCURRENCE(S)                       |
|%I0025 |_______________________________________________________|
|        | LEN |
                                   OFFLINE
C:\LM90\LESSON                    PRG: LESSON  BLK:  MAIN  SIZE:   155 RUNG 0004
REPLACE                  %I0003  :        ::
```

6. If an error occurs while replacing one of the reference addresses, the rung number where the error occurred is displayed at the top of the ladder diagram screen with the cursor on the erroneous item. In addition, a prompt displayed on the message line will ask you to either stop the function or skip the erroneous replacement and continue the function. Press **Y** (Yes) to skip the erroneous replacement and continue with the search and replacement function, or press **N** (No) to stop. A search and replace function, however, cannot be aborted.

Implicit Search

Implicit references are those references which are not directly programmed in the logic. However, due to the length of a function parameter, they are included. An implicit search enables you to locate these implicit references within a program.

Selecting **IMPLICIT** in the *Usage* field of the search function window means that both explicit **and** implicit references will be searched for. (Explicit references are those references you entered while programming the logic for your program or subroutine block.)

In the following example, %I0001 is an explicit reference and %I0002 through %I0016 are implicit references.

```
|%M0001    |------|                                                     %Q0001
|--| |-----| ADD  |-----------------------------------------------------( )-
|          | INT  |
|          |      |
|%I0001 ---|I1   Q|-%Q0017
|          |      |
| CONST ---|I2    |
| +0004    |______|
```

Search by Reference Type

The search by reference type function allows you to locate all uses of a specific reference type within the program. To initiate this type of search, enter the % character and the user reference (e.g., %R for register references) in the *Search for* field. Then, press the **Enter** key. The % character is required in order to distinguish a user reference from a name character.

Quick Search for a Coil

The quick search feature allows you to search for a coil with a particular reference address by entering the reference address on the command line and then pressing the **Search** (**F4**) softkey. The search will begin at the current location and continue to search for the reference address until a coil with the specified address is found. The reference address continues to be displayed on the command line so that you can find the next occurrence of a coil with the specified reference address by simply pressing **Search** (**F4**) again.

Section 12: Online Editing/Monitoring

In addition to offline editing, Logicmaster 90-30/20/Micro software supports several online functions. To make online changes, the programmer must be in **ONLINE** mode; and logic in the current folder and PLC must be **EQUAL**. Refer to chapter 8, "Program Utilities," for information on verifying the program with the PLC.

The functions available when online and equal include:

- Monitoring logic and registers.
- Forcing and overriding discrete references.
- Substituting instructions, constant values, and reference addresses.
- Online changes for data values.
- Inserting and editing rungs in **STOP** mode.

Note

Changes made when **ONLINE** and **EQUAL** are restricted to those of equal size. The PLC is updated as each change is completed.

Inserting or Editing Rungs (Block Edit)

Logic in the program folder and the PLC may be inserted or edited from the program editor while the PLC is in **STOP** mode and the programmer is **ONLINE** and **EQUAL**. After pressing **INSERT** (**F1**) or **EDIT** (**F2**) to enter **INSERT** or **EDIT** mode and making a change, the status line will change from **LOGIC EQUAL** to **BLOCK EDIT** One or more rungs may be inserted or modified. After accepting all the changes, the modified logic may be saved to both the program folder and PLC by pressing **ALT-S**.

ALT-S may also be used to save a single subroutine block while the PLC is in **STOP** mode.

Substitutions

If the programmer is in **ONLINE** mode and communicating with an operating PLC and if the program logic in the PLC and in the programmer are **EQUAL**, an instruction, constant, or reference address may be substituted with another. Every substitution updates the logic in the program folder and in PLC memory.

Instruction substitutions are permitted within the groups listed in the following table. Data types cannot be changed within each group; therefore, it is not necessary to include the data type.

Table 3-12. Substitution Groups

Function	Description	Type This
	Contacts	
−| |−	Normally open contact.	&NOCON
−|/|−	Normally closed contact.	&NCCON
	Coils	
−( )−	Normally open coil.	&NOCOI
−(/)−	Negated coil.	&NCCOI
−(S)−	SET coil.	&SL
−(R)−	RESET coil.	&RL
−(M)−	Retentive coil.	&NOMC
−(/M)−	Negated retentive coil.	&NCM
−(SM)−	Retentive SET coil.	&SM
−(RM)−	Retentive RESET coil.	&RM
−(↑)−	Positive transition coil.	&PCOI
−(↓)−	Negative transition coil.	&NCOI
	Retentive On-Delay Timer Function	
ONDTR_TENTHS	Tenth of a second time base.	&ON_TE
ONDTR_HUNDTHS	Hundredth of a second time base.	&ON_H
ONDTR_THSDTHS	Thousandth of a second time base.	&ON_TH
	On-Delay Timer Function	
TMR_TENTHS	Tenth of a second time base.	&TM_TE
TMR_HUNDTHS	Hundredth of a second time base.	&TM_H
TMR_THSDTHS	Thousandth of a second time base.	&TM_TH
	Off-Delay Timer Function	
TMR_TENTHS	Tenth of a second time base.	&OFDT_TE
TMR_HUNDTHS	Hundredth of a second time base.	&OFDT_H
TMR_THSDTHS	Thousandth of a second time base.	&OFDT_TH
	Counter Functions	
UPCTR	Up counter.	&UP
DNCTR	Down counter.	&DN
	Integer Math Functions	
ADD_INT	Signed integer addition.	&AD
SUB_INT	Signed integer subtraction.	&SUB
MUL_INT	Signed integer multiplication.	&MUL
DIV_INT	Signed integer division.	&DIV
MOD_INT	Signed integer modulo.	&MOD
	Double Integer Math Functions	
ADD_DINT	Double precision integer addition.	&AD_DI
SUB_DINT	Double precision integer subtraction.	&SUB_DI
MUL_DINT	Double precision integer multiplication.	&MUL_DI
DIV_DINT	Double precision integer division.	&DIV_DI
MOD_DINT	Double precision integer modulo.	&MOD_DI

Table 3-12. Substitution Groups (cont'd)

Function	Description	Type This
	Integer Comparison Functions	
EQ_INT	Test for equality between two integers.	&EQ
NE_INT	Test for no equality between two integers.	&NE
GT_INT	Test for one integer value greater than another.	&GT
GE_INT	Test for one integer value greater than or equal to another.	&GE
LT_INT	Test for one integer value less than another.	&LT
LE_INT	Test for one integer value less than or equal to another.	&LE
RANG_INT	Test the input value against a range of two numbers.	&RANG
	Double Integer Comparison Functions	
EQ_DINT	Test for equality between two double precision integers.	&EQ_DI
NE_DINT	Test for no equality between two double precision integers.	&NE_DI
GT_DINT	Test for one double precision integer value greater than another.	&GT_DI
GE_DINT	Test for one double precision integer value greater than or equal to another.	&GE_DI
LT_DINT	Test for one double precision integer value less than another.	&LT_DI
LE_DINT	Test for one double precision integer value less than or equal to another.	&LE_DI
RANG_DINT	Test the input value again st a range of two numbers.	&RANG_DI
	Bit Operation Functions (Words)	
AND_WORD	Logical "and" of two 16-bit word strings.	&AN
OR_WORD	Logical "or" of two 16-bit word strings.	&OR
XOR_WORD	Logical "exclusive or" of two 16-bit word strings.	&XO
	Shift Bit String Functions (Words)	
SHL_WORD	Shift bit string left.	&SHL
SHR_WORD	Shift bit string right.	&SHR
	Rotate Bit String Functions (Words)	
ROL_WORD	Rotate bit string left.	&ROL
ROR_WORD	Rotate bit string right.	&ROR
	Bit Set/Clear Functions (Words)	
BIT_SET_WORD	Set a bit within a bit string to 1.	&BS
BIT_CLR_WORD	Clear a bit within a bit string.	&BCL
	Integer Search Table Functions	
ARRAY_MOVE	Copy from one array to another.	&AR
SRCH_EQ	Search for array values equal to a specified value.	&SRCHE
SRCH_NE	Search for array values not equal to a specified value.	&SRCHN
SRCH_GT	Search for array values greater than a specified value.	&SRCHGT
SRCH_GE	Search for array values greater than or equal to a specified value.	&SRCHGE
SRCH_LT	Search for array values less than a specified value.	&SRCHLT
SRCH_LE	Search for array values less than or equal to a specified value.	&SRCHLE

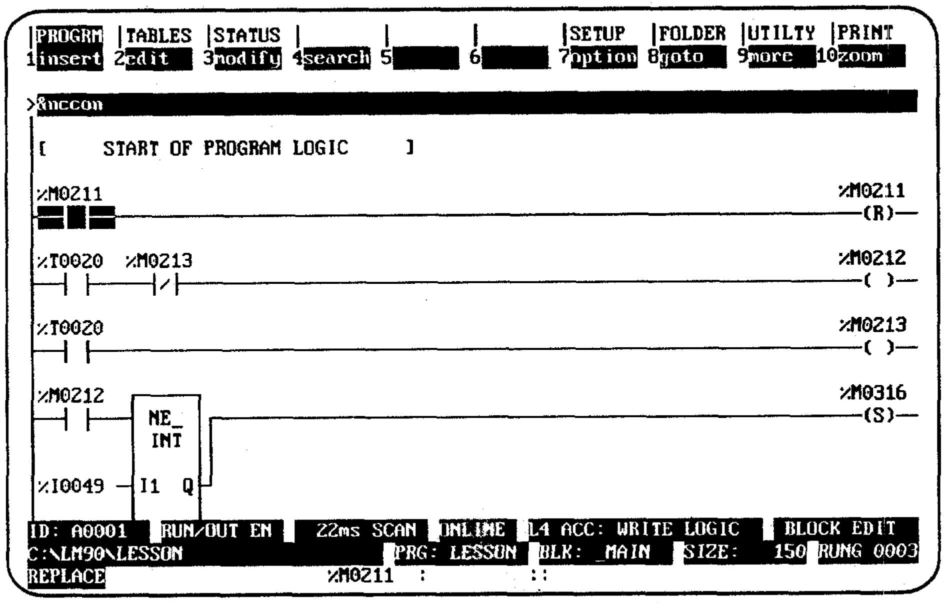
Table 3-12. Substitution Groups (cont'd)

Function	Description	Type This
	Double Integer Search Table Functions	
ARRAY_MOVE	Copy from one array to another.	&AR_DI
SRCH_EQ	Search for array values equal to a specified value.	&SRCHE_DI
SRCH_NE	Search for array values not equal to a specified value.	&SRCHN_DI
SRCH_GT	Search for array values greater than a specified value.	&SRCHGT_DI
SRCH_GE	Search for array values greater than or equal to a specified value.	&SRCHGE_DI
SRCH_LT	Search for array values less than a specified value.	&SRCHLT_DI
SRCH_LE	Search for array values less than or equal to a specified value.	&SRCHLE_DI
	Integer Search Table Functions (Byte)	
ARRAY_MOVE	Copy from one array to another.	&AR_BY
SRCH_EQ	Search for array values equal to a specified value.	&SRCHE_BY
SRCH_NE	Search for array values not equal to a specified value.	&SRCHN_BY
SRCH_GT	Search for array values greater than a specified value.	&SRCHGT_BY
SRCH_GE	Search for array values greater than or equal to a specified value.	&SRCHGE_BY
SRCH_LT	Search for array values less than a specified value.	&SRCHLT_BY
SRCH_LE	Search for array values less than or equal to a specified value.	&SRCHLE_BY
	Integer Search Table Functions (Words)	
ARRAY_MOVE	Copy from one array to another.	&AR_W
SRCH_EQ	Search for array values equal to a specified value.	&SRCHE_W
SRCH_NE	Search for array values not equal to a specified value.	&SRCHN_W
SRCH_GT	Search for array values greater than a specified value.	&SRCHGT_W
SRCH_GE	Search for array values greater than or equal to a specified value.	&SRCHGE_W
SRCH_LT	Search for array values less than a specified value.	&SRCHLT_W
SRCH_LE	Search for array values less than or equal to a specified value.	&SRCHLE_W

Modifying Instructions

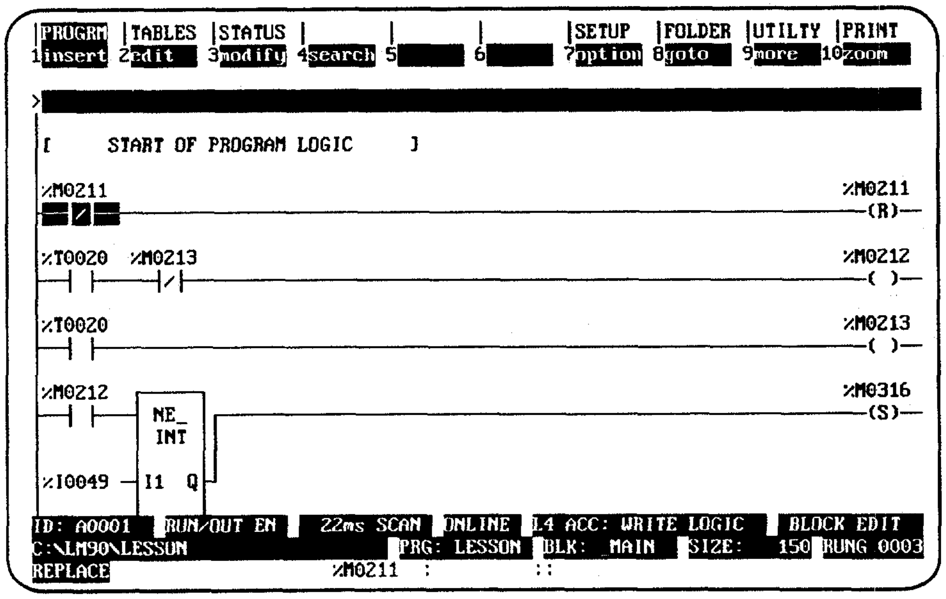
The following steps describe how to change relay ladder diagram elements and update the PLC while online and the PLC is running.

1. Place the cursor on the element to be changed.
2. Enter the new instruction mnemonic on the command line. The new instruction must be in the same instruction group as the existing instruction. (Instruction groups are listed in the table at the beginning of this section.)



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3modify 4search 5       6       7option 8goto   9more  10zoom
>&nccon
[     START OF PROGRAM LOGIC      ]
%M0211                                                              %M0211
%T0020  %M0213                                                      %M0212
%T0020                                                              %M0213
%M0212                                                              %M0316
         NE_
         INT
%I0049 --I1  Q
ID: A0001   RUN/OUT EN    22ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   BLOCK EDIT
C:\LM90\LESSON                      PRG: LESSON  BLK: _MAIN  SIZE:   150 RUNG 0003
REPLACE                           %M0211  :          ::
```

3. Press the **`Enter`** key, and then type `Y` (Yes) in response to the confirmation prompt.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3modify 4search 5       6       7option 8goto   9more  10zoom
>
[     START OF PROGRAM LOGIC      ]
%M0211                                                              %M0211
%T0020  %M0213                                                      %M0212
%T0020                                                              %M0213
%M0212                                                              %M0316
         NE_
         INT
%I0049 --I1  Q
ID: A0001   RUN/OUT EN    22ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   BLOCK EDIT
C:\LM90\LESSON                      PRG: LESSON  BLK: _MAIN  SIZE:   150 RUNG 0003
REPLACE                           %M0211  :          ::
```

The same steps are used to replace functions for other functions in the same instruction group. For example, the NE_INT function can be replaced with the LE_INT function. With the cursor on the NE_INT function, enter the new instruction mnemonic **&LE_INT** on the command line and press the **Enter** key.

Using the Modify Softkey

The **Modify (F3)** softkey enables you to substitute instructions from the rung editor environment by using the rung editor instruction softkeys. Position the cursor on the rung where the substitution is to be made and press **Modify** (**F3**). Then, use the function softkeys to enter the new instruction. You can also use the **Modify (F3)** softkey to change a reference address or constant parameter and to create/modify nicknames.

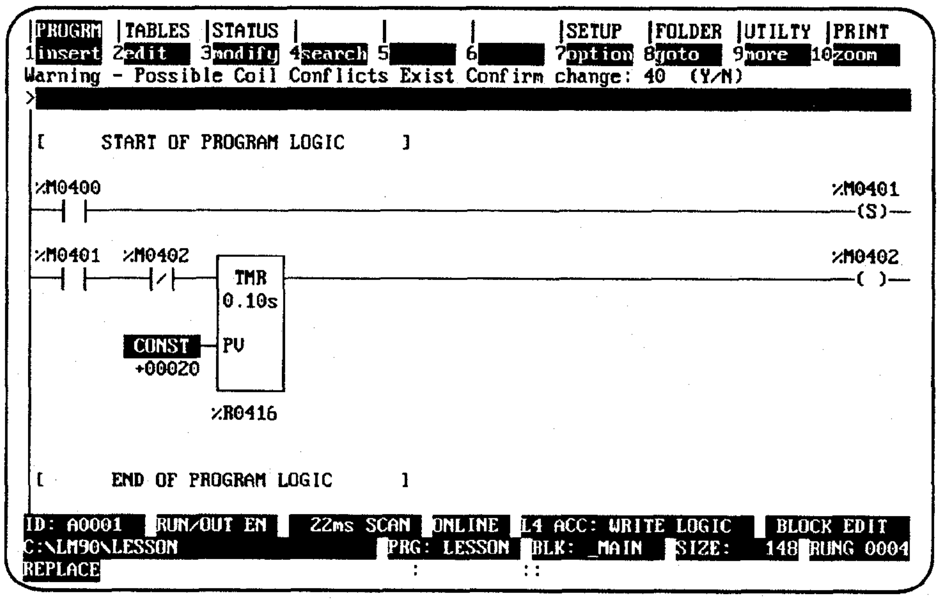
To restore the original values prior to the substitution, press **ALT-A**.

When the change is completed, press the **Escape** key.

Modifying a Reference Address or Constant

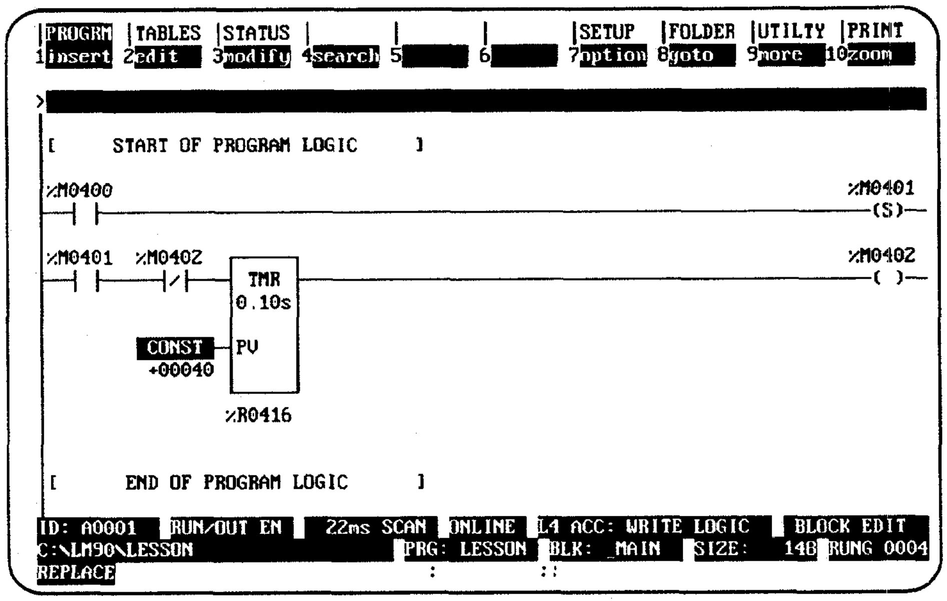
A reference address or other parameter can be changed while online and the PLC is running.

1. Place the cursor on the reference to be changed. In this example, the cursor is positioned on the TMR function's PV parameter, which currently contains a value of 20. Enter a new value of 40 on the command line, and press the **Enter** key.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3modify 4search 5       6       7option 8goto   9more  10zoom
Warning - Possible Coil Conflicts Exist Confirm change: 40  (Y/N)
>

[      START OF PROGRAM LOGIC      ]

%M0400                                                              %M0401
--| |---------------------------------------------------------------(S)--

%M0401  %M0402                                                      %M0402
--| |-----|/|---+-------+-------------------------------------------( )--
                | TMR   |
                |0.10s  |
                |       |
       CONST ---|PV     |
       +00020   |       |
                +-------+
                 %R0416


[       END OF PROGRAM LOGIC       ]
ID: A0001  RUN/OUT EN   22ms SCAN  ONLINE  L4 ACC: WRITE LOGIC    BLOCK EDIT
C:\LM90\LESSON                     PRG: LESSON  BLK: _MAIN  SIZE:   148 RUNG 0004
REPLACE                             :            ::
```

2. Press **Y** (Yes) in response to the confirmation prompt to update the PLC with the new operand value.


```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3modify 4search 5       6       7option 8goto   9more  10zoom
>
[     START OF PROGRAM LOGIC      ]

%M0400                                                              %M0401
--| |---------------------------------------------------------------(S)--

%M0401  %M0402                                                      %M0402
--| |------|/|---[ TMR  ]-------------------------------------------( )--
                 [0.10s ]
          CONST -[PV    ]
         +00040
                  %R0416

[      END OF PROGRAM LOGIC      ]
ID: A0001  RUN/OUT EN   22ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   BLOCK EDIT
C:\LM90\LESSON                     PRG: LESSON  BLK: _MAIN  SIZE:  148 RUNG 0004
REPLACE                                 :            ::
```

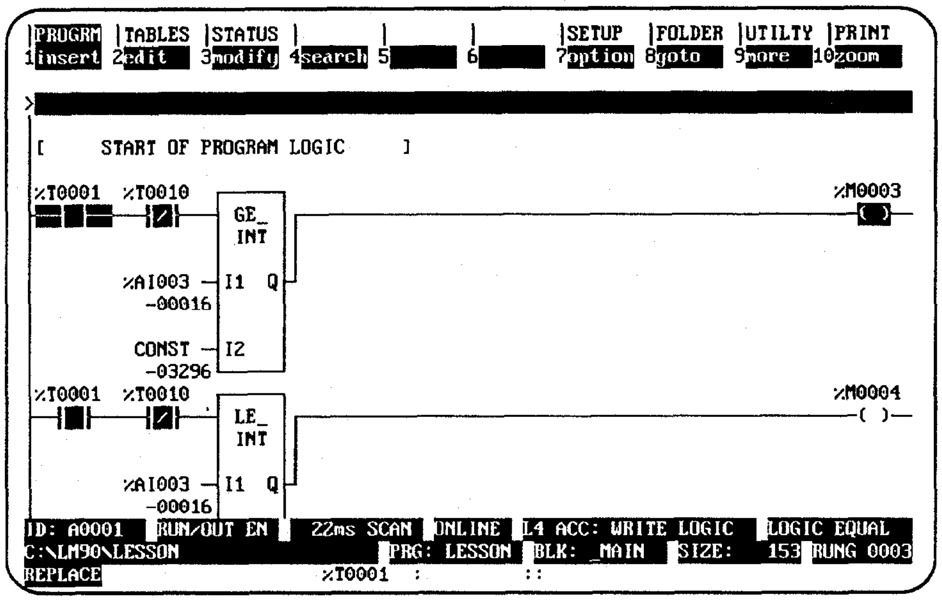
Forcing and Overriding Discrete References

A value that is overridden can be protected from change by the program logic; however, an overridden value can be forced or toggled.

1. Discrete points can be forced or toggled by positioning the cursor on a contact or coil with the reference address to be modified, and pressing **F12** or the keypad (–) key.

2. For input points that are being scanned or output points that the logic program changes, you must first override the point by pressing **F11** or the keypad (*) key before forcing the reference.
3. Once a point has been overridden, the first character of the reference address or nickname will flash.
4. To remove an override, position the cursor on the use of the reference address, and press **F11** or the keypad (*) key again.

To force the discrete reference %T0001 in the following example, position the cursor on an element which uses that reference. Then, press **F12** to update the PLC.

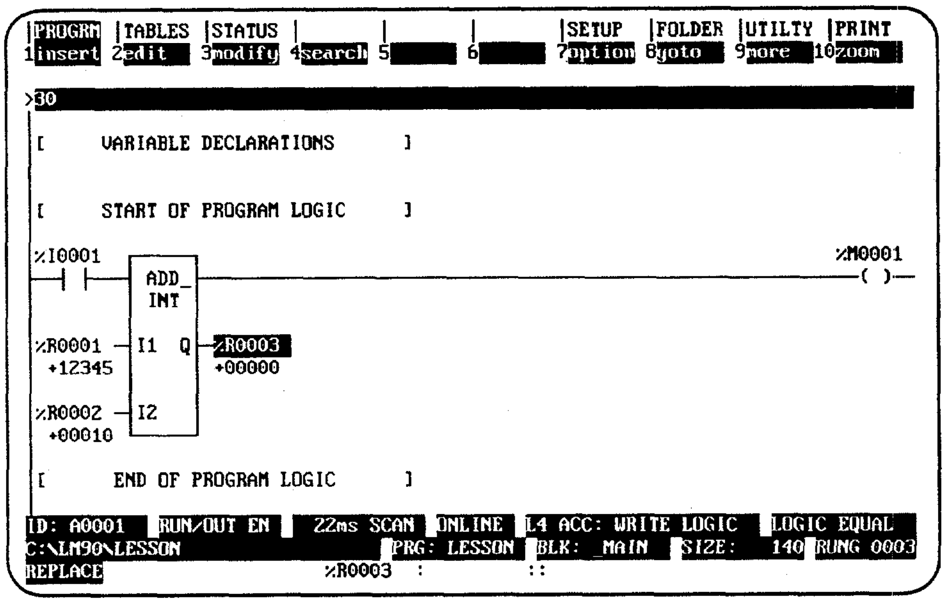


```
[     START OF PROGRAM LOGIC     ]
%T0001  %T0010                                        %M0003
        GE_ INT
%AI003 - I1 Q
-00016
CONST - I2
-03296
%T0001  %T0010                                        %M0004
        LE_ INT
%AI003 - I1 Q
-00016
ID: A0001  RUN/OUT EN  22ms SCAN  ONLINE  L4 ACC: WRITE LOGIC  LOGIC EQUAL
C:\LM90\LESSON              PRG: LESSON  BLK: _MAIN  SIZE:  153  RUNG 0003
REPLACE                     %T0001  :        ::
```

Changing Register Values

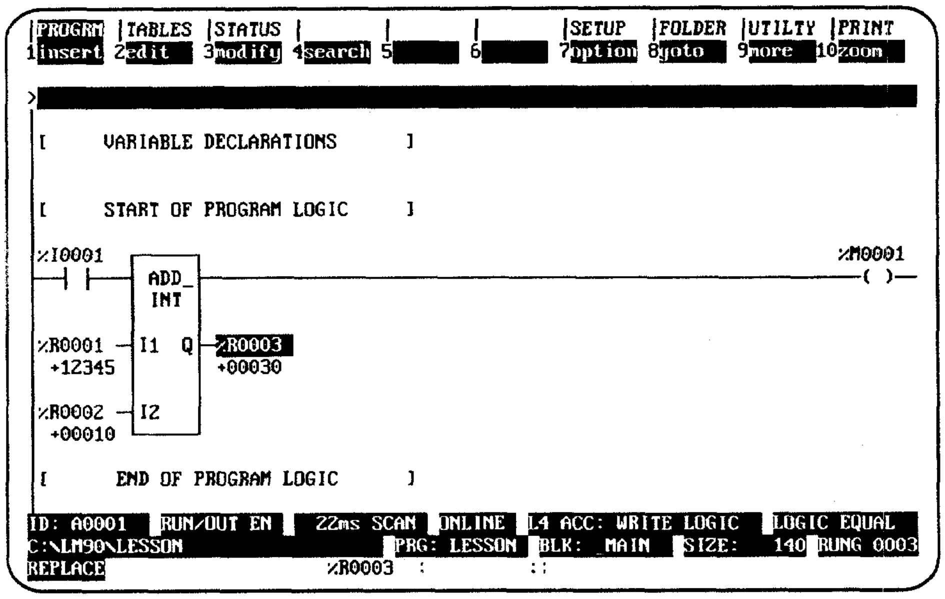
The following steps describe how to change the value of a register at the current cursor position.

1. Place the cursor on the register to be changed. (For this example, place the cursor on %R0003.)
2. Enter the new value on the command line. (For this example, enter **30** on the command line.)



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3modify 4search 5       6       7option 8goto   9more  10zoom
>30
[     VARIABLE DECLARATIONS       ]

[     START OF PROGRAM LOGIC      ]

%I0001                                                             %M0001
--| |---+ADD_+------------------------------------------------------( )--
        | INT|
%R0001 -|I1 Q|-%R0003
 +12345 |    | +00000
%R0002 -|I2  |
 +00010 +----+
[      END OF PROGRAM LOGIC       ]
ID: A0001  RUN/OUT EN   22ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                   PRG: LESSON  BLK: _MAIN  SIZE:   140 RUNG 0003
REPLACE                          %R0003  :          : :
```

3. Press the **Enter** key. The register value for this example will change from **0** to **30**, as shown in the screen below.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3modify 4search 5       6       7option 8goto   9more  10zoom
>
[     VARIABLE DECLARATIONS       ]

[     START OF PROGRAM LOGIC      ]

%I0001                                                             %M0001
--| |---+ADD_+------------------------------------------------------( )--
        | INT|
%R0001 -|I1 Q|-%R0003
 +12345 |    | +00030
%R0002 -|I2  |
 +00010 +----+
[      END OF PROGRAM LOGIC       ]
ID: A0001  RUN/OUT EN   22ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                   PRG: LESSON  BLK: _MAIN  SIZE:   140 RUNG 0003
REPLACE                          %R0003  :          : :
```

Chapter 4 Reference Tables

The display reference tables function is used to:

- Display tables of reference values.
- Set all the reference values in the table to zero.
- Change the formats in which reference tables are displayed or printed.
- Change reference values.
- Override discrete references (remove reference control from the PLC).

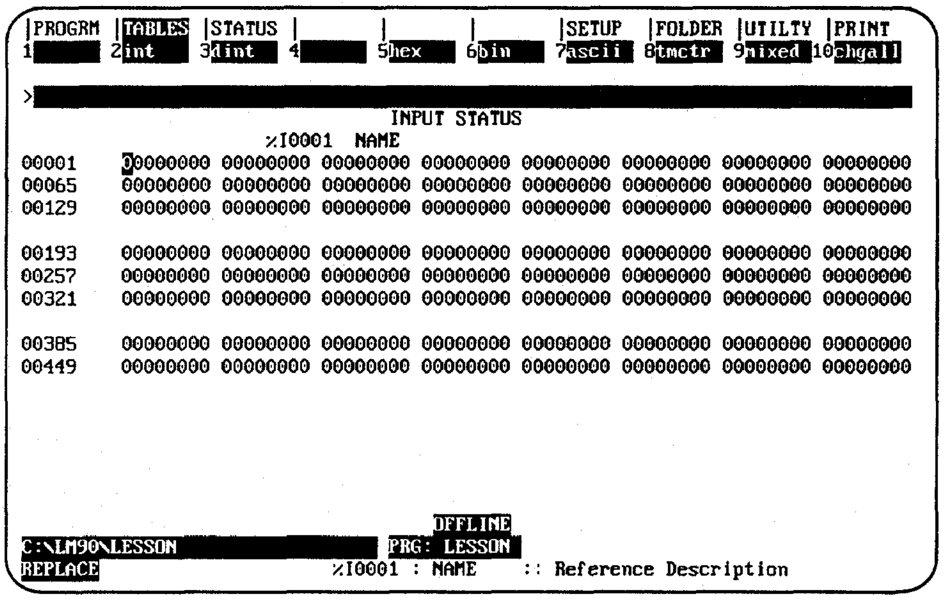
Chapter 4 contains the following sections:

Section	Title	Description	Page
1	Displaying Reference Tables	Explains how to display reference tables.	4-2
2	Changing Reference Table Values	Explains how to change reference values.	4-4
3	Using Overrides	Explains how to remove control of discrete references from the logic program.	4-9
4	Changing Display Formats	Explains how to make format changes.	4-11
5	Mixed Reference Tables	Describes mixed reference tables.	4-20

Section 1: Displaying Reference Tables

Reference tables can be displayed directly from the main menu in any programming mode, or from any main menu function screen. While a table is displayed, you can change the format of any references in the table for a particular application.

This is a sample reference table used for discrete inputs (%I):



```
|PROGRM |TABLES |STATUS |        |        |        |SETUP   |FOLDER  |UTILTY  |PRINT
1       2int    3dint   4        5hex     6bin     7ascii   8tmctr   9mixed   10chgall

>
                                     INPUT STATUS
                        %I0001  NAME
00001     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00065     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00129     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

00193     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00257     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00321     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

00385     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00449     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000




                                          OFFLINE
C:\LM90\LESSON                           PRG: LESSON
REPLACE                       %I0001 : NAME    :: Reference Description
```

Note

In **OFFLINE** mode, reference values from the current program folder are displayed. In **ONLINE** or **MONITOR** mode, references from the PLC are displayed.

Offline values from a reference table (values from the folder) can also be printed using the print function. These values will have the formats you set up on the display. To print values displayed online, the values must first be loaded from the PLC and then printed.

Displaying a Reference Table

To display a reference table:

1. Enter one of the reference types listed in the following table:

User Reference	Reference Table
%I	Discrete input.
%Q	Discrete output.
%M	Discrete internal.
%T	Discrete temporary.
%G	Discrete Genius global data.
%S, %SA, %SB, %SC	Discrete system.
%AI	Analog input.
%AQ	Analog output.
%R	Register.

   A. To view the lowest-numbered table of a particular type, enter just the reference type. For example, for the lowest-numbered analog input table, enter **`%AI`**.

   B. To view a table containing a specific reference, enter the reference or its nickname. For example, %AI123 or 123AI.

2. Then, press **`Tables`** (**`Shift-F2`**).
3. Once in the reference tables function, you may go to a different reference table by entering the reference on the command line and pressing the **`Enter`** key.

Using the Cursor to Select a Reference Table

You may go directly to the reference table of a reference (e.g., of a function block operand, contact, coil, etc.) under the cursor by pressing **`ALT-F2`**. Then, press **`Shift-F1`** from the reference table to return to the same place in the program.

Moving the Cursor in a Reference Table

Follow these guidelines for moving the cursor in a reference table:

- Use the cursor keys to move one value horizontally or vertically in the table.
- To move the cursor to a specific reference in the same table, enter the reference on the command line and press the **`Enter`** key. To move to a specific reference in any table, enter the reference in the command line and press the **`Enter`** key, or use **`Shift-F2`**.
- To display the next screen of the same table, use the Page Down key. Use the Page Up key to display the previous screen.
- Use the CTRL-Page Down keys to move the cursor to the first reference of the next line. Use the CTRL-Page Up keys to move the cursor to the first reference of the previous line.
- Use the Home key to go to the first value in the table. Use the End key to go to the last value in the table.

Section 2: Changing Reference Table Values

You can force (change) both discrete and register reference values with the reference tables function.

Note

If the programmer mode is **`OFFLINE`**, changes will only be made to the current program folder. Changes made to the program folder may later be stored to the PLC by using the store utility function, described in chapter 8, "Program Utilities." If the programmer mode is **`ONLINE`**, any changes will only be made to reference values in the PLC. Reference values in the PLC may then be loaded to the program folder by using the load utility function, also described in chapter 8. No changes to the PLC can be made while the programmer is in **`MONITOR`** mode.

Warning

Improper use of online program changes can damage equipment or cause personal injury.

Online program changes should always be made with extreme care. Online changes can have serious and unforeseen results on a control system, and on the process to which it applies, if they are improperly used. It is recommended that these functions not be used with people near the equipment. If possible, they should be done with direct visual control over the system and the process. Proper external power disconnects should be made to prevent undesired equipment operation.

In order to make online changes, the status line at the bottom of the screen must show that the programmer is online to the CPU.

Changing a Register Reference

You can enter or change the values of a register reference as described below. This method can be used to load a value into a register. To change a register reference value:

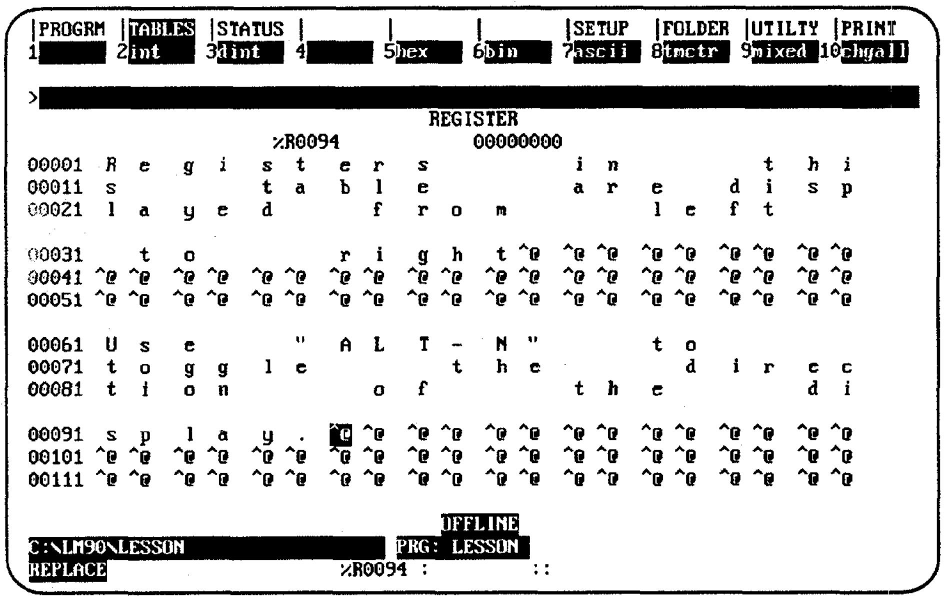
1. In the reference table that contains the reference, move the cursor to highlight the reference or enter the reference on the command line.
2. Enter a new value on the screen's command line and press the **`Enter`** key. The new value will be interpreted in the format (for example, signed integer) currently assigned to the reference.

ASCII String Entry

An ASCII string up to 79 characters, the size of the command line, can be entered in word-oriented tables. The string of text is entered the same as a name explanation and is displayed one character at a time, beginning with the reference the cursor is on.

The string is entered by typing the text, enclosed in quotes, on the command line and then pressing the **`Enter`** key. To include quotes within a string of text, you must begin the string with two colons. For example, to enter the string: **`Use "ALT-N" to toggle`** you must enter **`::Use "ALT-N" to toggle`** on the command line. A string can be accepted without a closing quote, but only to the last non-space character. Trailing spaces are lost without the closing quote.

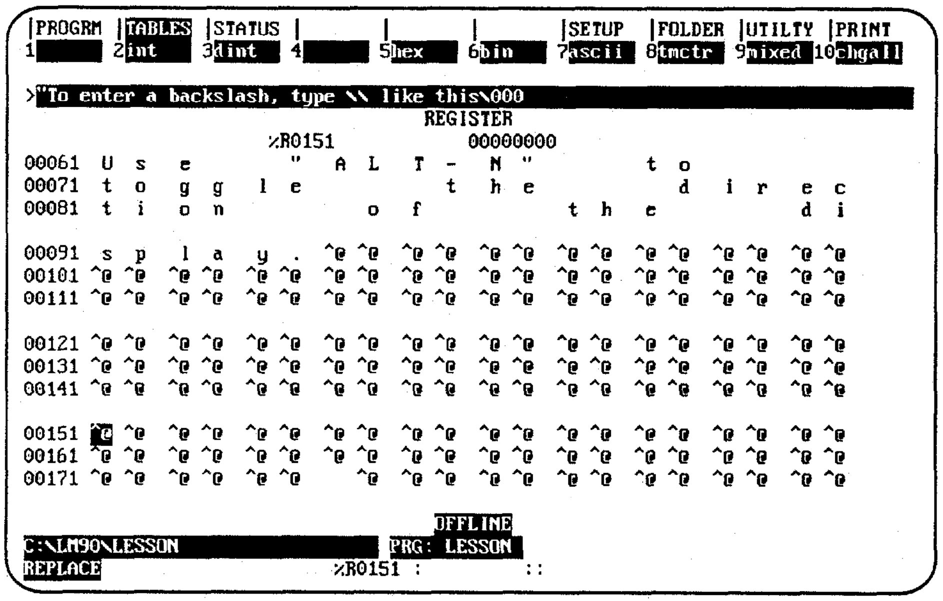
The following screen shows the reference table display after the string "Registers in this table are displayed from left to right" is entered. (The display mode of any reference table can be changed from right to left or left to right, as explained on the following pages of this manual.)



```
|PROGRM |TABLES |STATUS |        |        |        |SETUP   |FOLDER  |UTILTY  |PRINT
1       2int    3dint   4        5hex     6bin     7ascii   8tmctr   9mixed  10chgall

>
                                            REGISTER
                        %R0094                  00000000
00001 R  e   g  i   s  t   e  r   s                  i  n                 t   h  i
00011 s                s  t   a   b  l   e            a  r   e      d  i   s  p
00021 l  a   y  e   d             f   r  o   m                l  e   f  t

00031    t   o                r  i   g  h   t  ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@
00041 ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@
00051 ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@

00061 U  s   e          "  A  L   T  -   N  "              t  o
00071 t  o   g  g   l  e              t   h  e                 d   i  r   e  c
00081 t  i   o  n             o   f               t  h   e             d  i

00091 s  p   l  a   y  .   ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@
00101 ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@
00111 ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@

                                                OFFLINE
C:\LM90\LESSON                           PRG: LESSON
REPLACE                            %R0094 :            ::
```

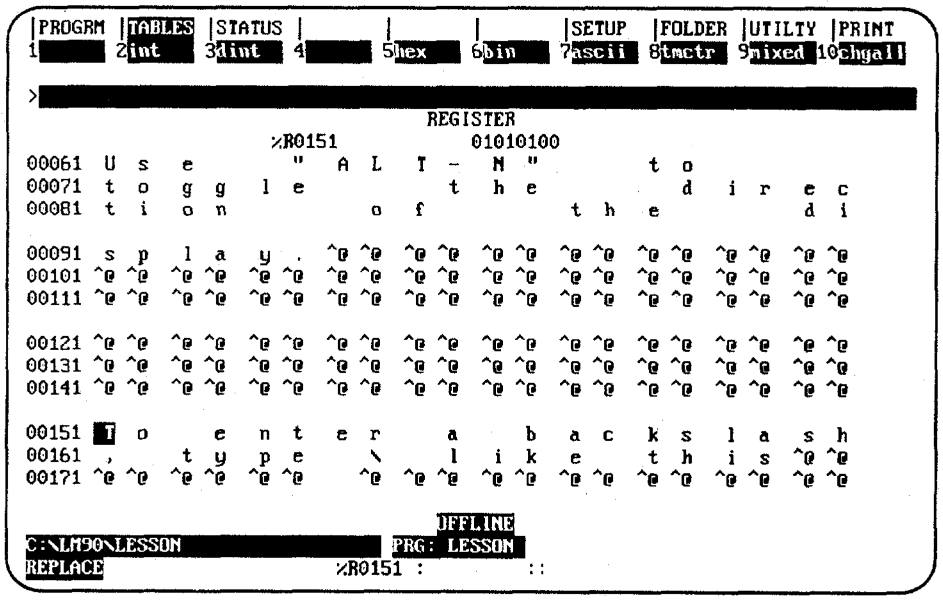
Non-printable characters (e.g., "null" to terminate a string) may be included in the ASCII string by entering a backslash and the three-digit decimal value of the non-printable character (e.g., "\000"). If you wish to include a backslash (\) in the text, enter two backslashes. There is no limit, other than the size of the command line, on the number of non-printable characters a string may have. The software performs a check to verify that the value is a valid decimal number.

This example shows a null terminated string.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1       2int    3dint   4       5hex    6bin    7ascii  8tmctr  9mixed 10chgall

>"To enter a backslash, type \\ like this\000
                                      REGISTER
                      %R0151                00000000
00061 U  s   e         "   A  L   T  -   N  "          t  o
00071 t  o   g  g   l  e             t   h  e             d   i  r   e  c
00081 t  i   o  n             o   f             t  h   e             d  i

00091 s  p   l  a   y  .   ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@
00101 ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@
00111 ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@

00121 ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@
00131 ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@
00141 ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@

00151 ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@
00161 ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@
00171 ^@ ^@  ^@ ^@  ^@ ^@     ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@

                                          OFFLINE
C:\LM90\LESSON                      PRG: LESSON
REPLACE                        %R0151 :           ::
```

The next example shows the reference table after the null terminated string is accepted.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1       2int    3dint   4       5hex    6bin    7ascii  8tmctr  9mixed 10chgall

>
                                      REGISTER
                      %R0151                01010100
00061 U  s   e         "   A  L   T  -   N  "          t  o
00071 t  o   g  g   l  e             t   h  e             d   i  r   e  c
00081 t  i   o  n             o   f             t  h   e             d  i

00091 s  p   l  a   y  .   ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@
00101 ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@
00111 ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@

00121 ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@
00131 ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@
00141 ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@

00151 T  o      e   n  t   e  r      a      b   a  c   k  s   l  a   s  h
00161 ,      t  y   p  e      \      l   i  k   e      t  h   i  s   ^@ ^@
00171 ^@ ^@  ^@ ^@  ^@ ^@     ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@  ^@ ^@

                                          OFFLINE
C:\LM90\LESSON                      PRG: LESSON
REPLACE                        %R0151 :           ::
```

Changing the Reference Display Mode (ALT-N)

The ALT-N key sequence enables you to display table data either right to left (lowest reference address on the right) or left to right (lowest reference address on the left). The display mode can be changed in any table (fixed or mixed) and in any program mode (**OFFLINE**, **ONLINE**, or **MONITOR**).

The Home End, Page Up and Page Down, Previous, Next, and cursor keys function the same, regardless of the view mode selected.

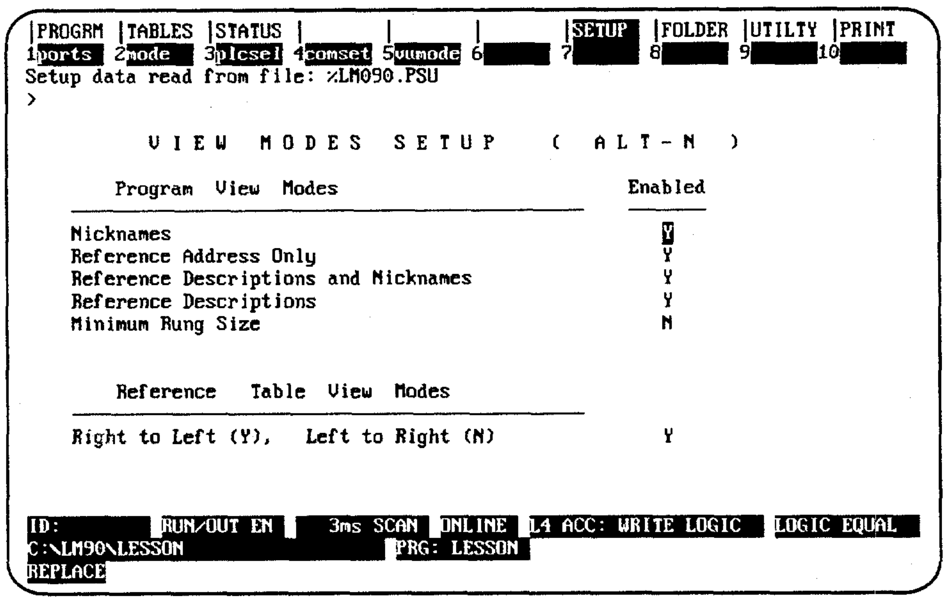
Data displayed in ASCII format differs slightly, in that the quotes are removed and the format is treated more as a byte (1 character) instead of a word (2 characters).

Note

The print function cannot distinguish between these two display modes and prints all tables right to left.

The default display mode is the last mode selected with ALT-N. If the display mode has never been changed, the default set in the programmer setup is used (see chapter 6, "Programmer Setup")., Follow the steps below to change the display mode.

1. The default reference table display mode is displayed on the View Modes Setup screen. To display this screen, press **Setup** (**F7**) from the Programming Software main menu and then **View Modes Setup** (**F5**) from the Programmer Setup menu. The view mode in the following screen is right to left (the default display).



```
|PROGRM |TABLES |STATUS |      |      |      |SETUP |FOLDER |UTILITY |PRINT
1ports  2mode   3plcsel 4comset 5vumode 6     7      8      9       10
Setup data read from file: %LM090.PSU
>

        V I E W   M O D E S   S E T U P    ( A L T - N )

        Program  View  Modes                              Enabled

     Nicknames                                              Y
     Reference Address Only                                 Y
     Reference Descriptions and Nicknames                   Y
     Reference Descriptions                                 Y
     Minimum Rung Size                                      N

        Reference   Table  View  Modes

     Right to Left (Y),   Left to Right (N)                 Y

ID:           RUN/OUT EN    3ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                        PRG: LESSON
REPLACE
```

2. To change the default view mode to display left to right, move the cursor to the, *Enabled,* field for the, *Reference Table View Modes,* entry. Then, use the Tab key to toggle the selection to **N** (No) or enter **N**.

3. To validate and save the default view mode selection in the home directory file %LM090.PSU, press **ALT-U**, the **Escape** key, or the shifted function key.

Forcing a Discrete Reference

In discrete reference tables, a reference can be forced on or off. If the reference being forced is currently overridden (described below), it retains its new status until forced again. If the reference is not overridden, it retains its new status until changed by some other function, such as rung solution or I/O servicing. This usually occurs within one sweep.

You can enter the desired state (0 or 1) on the command line, or toggle the reference state as described below. The following method is easier.

To toggle a reference:

1. With the table displayed on the screen, place the cursor on the reference to be forced.
2. Press the keypad minus (**-**) key or **F12** key to change that reference to its opposite state. All logic elements in the program that use the reference will reflect the new status.

Changing the Values of a Word of Discrete References

You can change the value of the word where the cursor is. To do that, first you must change the number base of the references. For example, suppose you want to convert discrete references (see below) 0225 through 0240 to ones.

```
                      References 0225 through 0240
                               _______|_______
                              |               |
0256 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Follow the steps below to change a word of discrete references.

1. The cursor indicates the rightmost (lower-numbered) byte of the word to be changed. (The software automatically adjusts the cursor position if it is incorrectly placed.) Use the function keys to change the number base to one of the following: signed integer (by pressing **F2**) or hexadecimal (by pressing **F5**).
2. On the command line, type the equivalent of the binary value in the selected format.
3. Press the **Enter** key to change the reference at the cursor position to the new value.
4. Press **Binary** (**F6**) to convert the base back to binary values.

```
                      References 0225 through 0240
                               _______|_______
                              |               |
0256 00000000 00000000 11111111 11111111 00000000 00000000 00000000 00000000
```

Section 3: Overrides

Discrete references in Models 331 or higher CPU can be overridden from the reference tables display. (This feature is not available in Models 323, 321, 311, 313, 211, and Micro CPUs.)

Note

If you attempt to use overrides with a CPU below Model 331, you will see an error message on your screen stating that overrides are not allowed.

Discrete references that have been overridden are indicated by flashing digits on the reference table screen. An override removes control of the reference from its normal source. Overridden inputs ignore information from the devices wired to the I/O structure, such as limit switches or pushbuttons. Similarly, overridden outputs ignore programmed logic and internal power flow. Overrides are retained even when power is removed from the system. Non-relay functions such as timers, counters, math functions, and data move functions still work when a coil is overridden.

Warning

If overrides are applied to a reference associated with a transitional coil, the coil may pulse on for one sweep when the override is removed.

The override is a very powerful tool for program checking and maintenance. You can test a program in a PLC that is not connected to I/O hardware by using overrides to simulate inputs. You can also check a program when I/O is connected, by using overrides to prevent coil operation.

After the I/O is wired up, it can be tested by activating each coil with an override to verify I/O communications, module operation, power to a device, wiring to a device, indicator lights, fuses, and other hardware.

After the control system is thoroughly checked and placed in operation, the override is useful in a monitored system. If a sensor or input module should fail while the process is in operation, that input can be overridden. Thus, the process can be continued until it can be shut down safely.

References should not be overridden when the programmer is removed from the process, or when making copies of a program. Use the reference tables function to verify all inputs and coils before removing them from the programmer, or copying the program.

Using Overrides

Overrides should be used on an operating system only with extreme care.

Warning

Improper use of the override can damage equipment or cause personal injury.

1. Place the cursor on the reference to be overridden.

Caution

The reference will be overridden throughout the program, not just at the cursor location.

2. Press the keypad Asterisk (*) key or **F11**. This toggles the state of the reference between overridden and not overridden. When overridden is selected as the state of reference, the value displayed in the table will flash.

Removing Overrides

All discrete reference bits on a reference table screen, or starting character in a tag name on a displayed rung, will flash if they have been overridden. To remove an override from one reference, toggle it by placing the cursor on the reference and pressing the keypad Asterisk (*) key or **F11**.

To remove all overrides shown on the current screen, including the last three lines which may be hidden beneath the status lines (Press **ALT-E** to remove the status lines.):

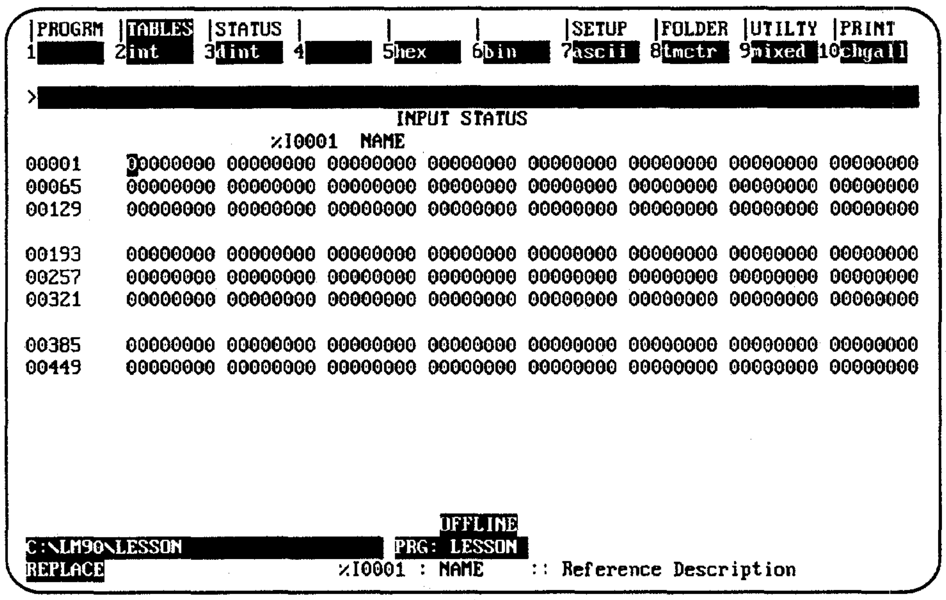
1. Press **Change All** (**F10**).
2. Press the keypad Asterisk (*) key or **F11**. The screen prompts: "Remove overrides from displayed references ? (Y/N)".
3. Enter **Y** (Yes) to remove the overrides.

Section 4: *Changing Display Formats*

The format of any reference in a table can be changed for a particular application while the table is displayed on the programmer screen. This section explains how to make format changes.

Discrete Reference Tables

The sample reference table below is used for discrete inputs (%I), discrete outputs (%Q), discrete internal coils (%M), discrete temporary coils (%T), discrete system status references (%S, %SA, %SB, and %SC), and discrete global data (%G).



```
|PROGRM |TABLES |STATUS |        |        |        |SETUP  |FOLDER |UTILTY |PRINT
1       2int    3dint   4        5hex     6bin     7ascii  8tmctr  9mixed 10chgall

>
                                 INPUT STATUS
                      %I0001  NAME
00001     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00065     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00129     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

00193     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00257     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00321     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

00385     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00449     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000




                                          OFFLINE
C:\LM90\LESSON                       PRG: LESSON
REPLACE                   %I0001 : NAME      :: Reference Description
```

The default format for discrete reference table displays is to have the values displayed or printed in binary. The format may be changed to signed integer or hexadecimal.

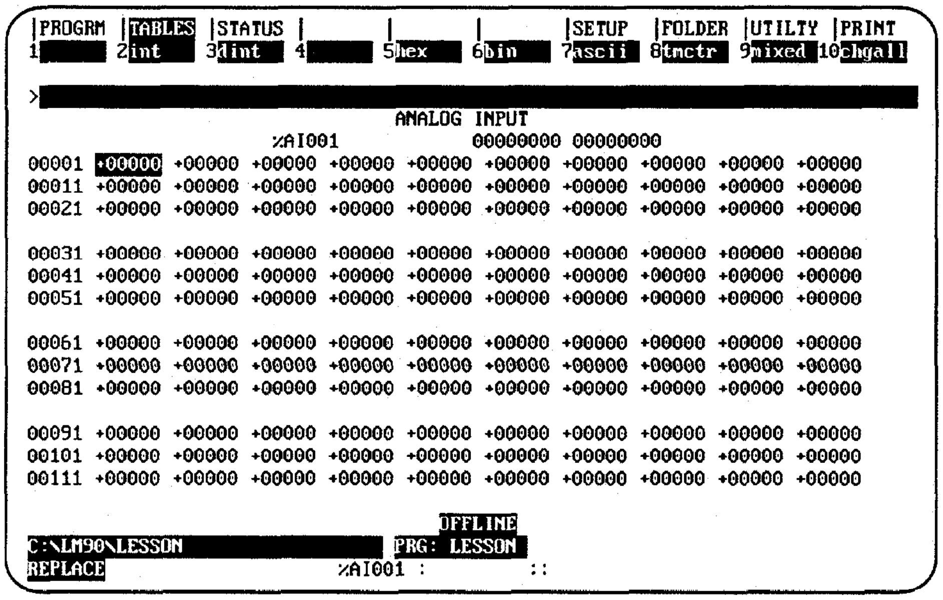
Any of these formats can be used for some of the values in the table, or for the entire table. Thus, each reference table can be uniquely formatted to be most meaningful for the type of information it contains. (Double precision integer and timer/counter are not allowed for discrete references.)

Note

The format for system status references (%S, %SA, %SB, and %SC) cannot be changed. System status references can only be displayed in binary format. The %S reference table values cannot be cleared or changed. %S memory is read only and cannot be written. The bit values in the %SA, %SB, and %SC reference tables can, however, be cleared or toggled.

Register Reference Tables

The sample register reference table below is used for system registers (%R), analog inputs (%AI), and analog outputs (%AQ).



```
|PROGRM |TABLES |STATUS |        |       |       |SETUP  |FOLDER |UTILTY |PRINT
1       2int    3dint   4        5hex    6bin    7ascii  8tmctr  9mixed  10chgall
>
                                  ANALOG INPUT
                  %AI001              00000000 00000000
00001 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000
00011 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000
00021 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000

00031 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000
00041 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000
00051 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000

00061 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000
00071 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000
00081 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000

00091 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000
00101 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000
00111 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000

                                           OFFLINE
C:\LM90\LESSON                        PRG: LESSON
REPLACE                     %AI001 :           ::
```

The default format for register reference table displays is for the values to be shown or printed as signed integers. The format can be changed to double precision integer, hexadecimal, binary, ASCII, or timer/counter. The format can also be changed for some of the values in the table or for the entire table.

In addition, the value of the reference at the cursor is displayed at the top of the table in binary format.

Display Formats

Use the function keys to change display formats. Changes are automatically saved when you leave a reference table display. The table will continue to be displayed in the new format until the format is changed again. If you want to save changes to the program folder **without** leaving the reference table display, press **ALT-U**.

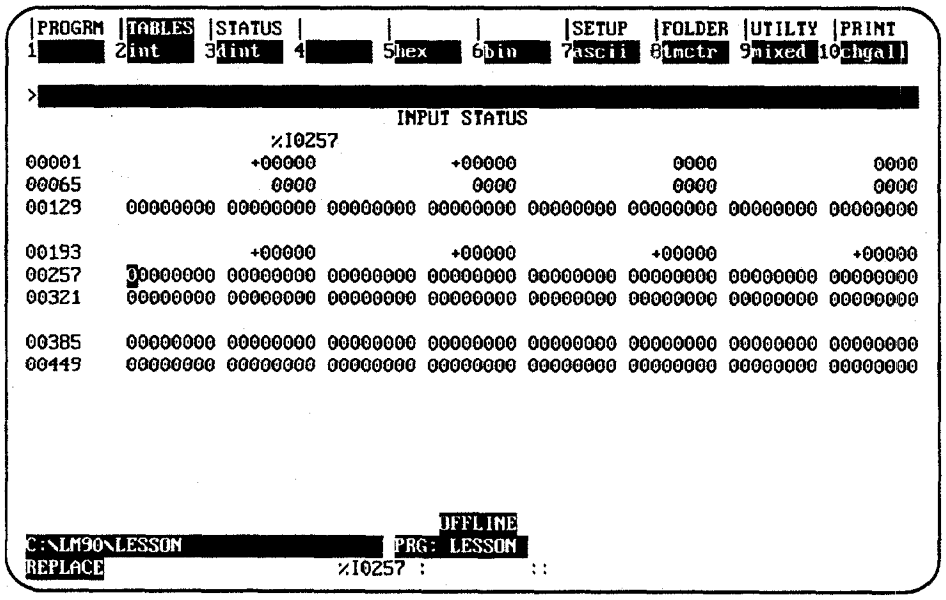
Format	Function Key	Description
Signed Integer	F2	A 16-bit number from −32,768 to +32,767. In discrete reference tables, 16 consecutive references are used for the display, beginning at a multiple-of-8-plus-one boundary (e.g., 1, 9, 17, 25, 33, etc.). In register reference tables, a single reference is required (e.g., −21846).
Signed Double Integer	F3	A 32-bit number from −2,147,483,648 to +2,147,483,647. This format can be used for register reference tables only. Two consecutive references are required, beginning at a multiple-of-2-minus-one boundary (e.g., 1, 3, 5, 7, etc.).
Hexadecimal	F5	A 16-bit number from 0000 to FFFF. When entering hexadecimal numbers on the command line, a leading A through F must be preceded by a zero. In addition, type an ***h*** at the end of the numeric string to ensure that the entry is read as a hexadecimal number. In discrete reference tables, 16 consecutive references are used, beginning at a multiple-of-16 boundary (e.g., 1, 17, 33, etc.). In register reference tables, a single reference is required (e.g., 0AAAAh).
Binary	F6	A 1-bit number with a value of 0 or 1. This format can be used for discrete reference tables only. A single reference is used. Binary values are displayed in groups of 8 references (e.g., 10101010). Register references cannot be displayed in binary format. However, the 16-bit binary equivalent of the value indicated by the cursor is automatically displayed at the top of the table.
ASCII	F7	8-bit encoded characters. This format can only be used in register reference tables. A single reference is required to make up 2 (packed) ASCII characters. The rightmost character of the pair corresponds to the low byte of the reference word. Bit 8 of each pair is a parity bit and is ignored. The remaining 7 bits in each section are converted as shown below. Command codes and non-displayable characters appear on the screen as the characters ^@ (e.g., X W).
TMRCTR	F8	Three contiguous reference addresses, which support timer/counter functions, display the current value, the preset value, and the control word. Regardless of whether the reference is assigned to a timer or counter, the format can be used as long as the reference type is %R. More information on this format is provided later in this chapter.
Mixed	F9	Select a user-defined table. Up to 99 user-defined tables may be created.
Change All	F10	Press **F10** to change the format of all elements in the table.

Table 4-1. ASCII Characters

Bit Pattern	Character	Bit Pattern	Character	Bit Pattern	Character	Bit Pattern	Character
X0000000	^@	X0100000	(blank)	X1000000	@	X1100000	′
X0000001	^A	X0100001	]	X1000001	A	X1100001	a
X0000010	^B	X0100010	″	X1000010	B	X1100010	b
X0000011	^C	X0100011	#	X1000011	C	X1100011	c
X0000100	^D	X0100100	$	X1000100	D	X1100100	d
X0000101	^E	X0100101	%	X1000101	E	X1100101	e
X0000110	^F	X0100110	&	X1000110	F	X1100110	f
X0000111	^G	X0100111	′	X1000111	G	X1100111	g
X0001000	^H	X0101000	(	X1001000	H	X1101000	h
X0001001	^I	X0101001	)	X1001001	I	X1101001	i
X0001010	^J	X0101010	*	X1001010	J	X1101010	j
X0001011	^K	X0101011	+	X1001011	K	X1101011	k
X0001100	^L	X0101100	,	X1001100	L	X1101100	l
X0001101	^M	X0101101	-	X1001101	M	X1101101	m
X0001110	^N	X0101110	.	X1001110	N	X1101110	n
X0001111	^O	X0101111	/	X1001111	O	X1101111	o
X0010000	^P	X0110000	0	X1010000	P	X1110000	p
X0010001	^Q	X0110001	1	X1010001	Q	X1110001	q
X0010010	^R	X0110010	2	X1010010	R	X1110010	r
X0010011	^S	X0110011	3	X1010011	S	X1110011	s
X0010100	^T	X0110100	4	X1010100	T	X1110100	t
X0010101	^U	X0110101	5	X1010101	U	X1110101	u
X0010110	^V	X0110110	6	X1010110	V	X1110110	v
X0010111	^W	X0110111	7	X1010111	W	X1110111	w
X0011000	^X	X0111000	8	X1011000	X	X1111000	x
X0011001	^Y	X0111001	9	X1011001	Y	X1111001	y
X0011010	^Z	X0111010	:	X1011010	Z	X1111010	z
X0011011	^[	X0111011	;	X1011011	[	X1111011	{
X0011100	^	X0111100	<	X1011100	\	X1111100	|
X0011101	^]	X0111101	=	X1011101	]	X1111101	}
X0011110	^^	X0111110	>	X1011110	^	X1111110	tilde
X0011111	^_	X0111111	?	X1011111	_	X1111111	delta

Changing the Display Format

To change the format of one reference, move the cursor to that reference and press the appropriate function key. The new format will appear at the rightmost position of the reference(s). Within any standard reference table, you can change as many reference formats as you want, as shown by the example below.

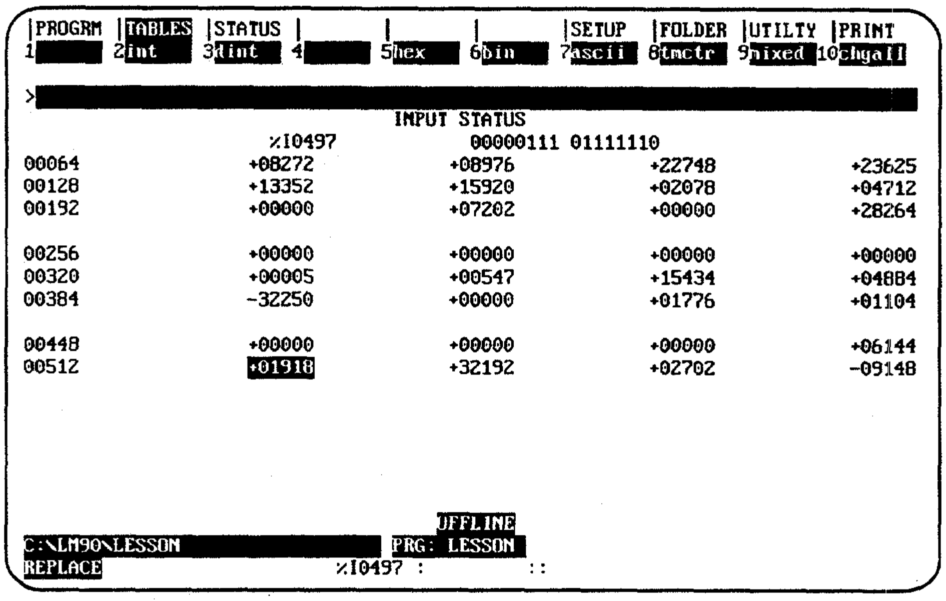


```
|PROGRM |TABLES |STATUS |        |        |        |SETUP  |FOLDER |UTILTY |PRINT
1       2int    3dint   4        5hex     6bin     7ascii  8tmctr  9mixed  10chgall
>
                                  INPUT STATUS
                %I0257
00001           +00000            +00000             0000              0000
00065             0000              0000             0000              0000
00129  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

00193           +00000            +00000           +00000            +00000
00257  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00321  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

00385  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00449  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000





                                        OFFLINE
C:\LM90\LESSON                          PRG: LESSON
REPLACE                    %I0257 :               ::
```

Changing the Format of a Table

You can also change the format of an entire table, including the values not currently on the screen or portions of the table. To change the format of an entire table:

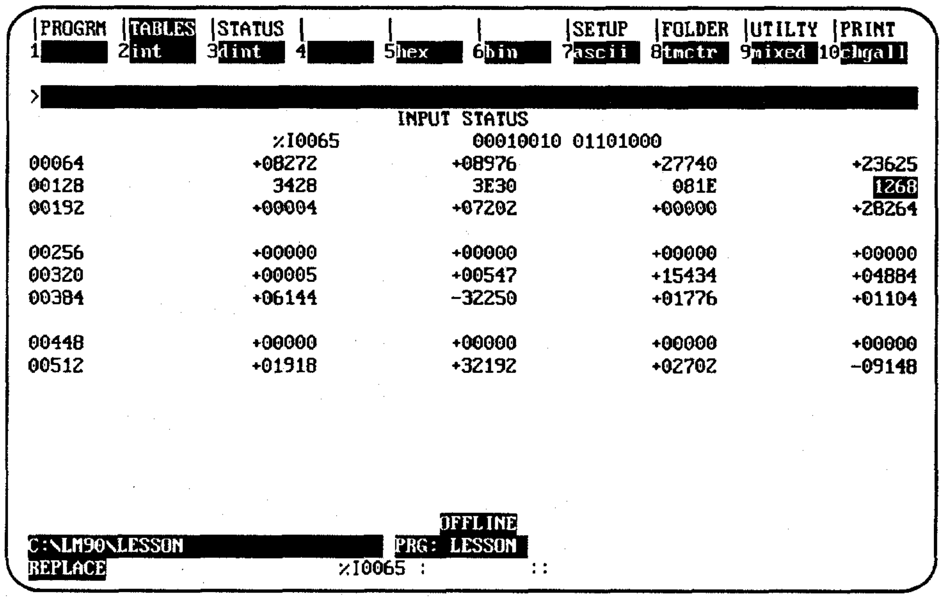
1. Press **Change All** (**F10**).
2. Select the format for the display. For example, press **F2** to select signed integer.



```
|PROGRM |TABLES |STATUS |        |        |        |SETUP  |FOLDER |UTILTY |PRINT
1       2int    3dint   4        5hex     6bin     7ascii  8tmctr  9mixed  10chgall
>
                                  INPUT STATUS
                %I0497            00000111 01111110
00064           +08272            +08976             +22748            +23625
00128           +13352            +15920             +02078            +04712
00192           +00000            +07202             +00000            +28264

00256           +00000            +00000             +00000            +00000
00320           +00005            +00547             +15434            +04884
00384           -32250            +00000             +01776            +01104

00448           +00000            +00000             +00000            +06144
00512           +01918            +32192             +02702            -09148





                                        OFFLINE
C:\LM90\LESSON                          PRG: LESSON
REPLACE                    %I0497 :               ::
```

The **Change All** (**F10**) key can also be used to change a specified block of range formats.

1. Enter a starting reference (the lower address) and an ending reference (the higher address) from the current table (e.g., 65i 128i).
2. Press **Change All** (**F10**).
3. Select the format (e.g., hex).



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1       2int    3dint   4       5hex    6bin    7ascii  8tmctr  9mixed  10chgall

>
                              INPUT STATUS
                %I0065          00010010 01101000
00064           +08272          +08976          +27740          +23625
00128             3428            3E30            081E            1268
00192           +00004          +07202          +00000          +28264

00256           +00000          +00000          +00000          +00000
00320           +00005          +00547          +15434          +04884
00384           +06144          -32250          +01776          +01104

00448           +00000          +00000          +00000          +00000
00512           +01918          +32192          +02702          -09148



                                          OFFLINE
C:\LM90\LESSON                          PRG: LESSON
REPLACE                  %I0065 :               ::
```

Entering only a valid starting reference will cause the formats from the starting reference to the end of the table to change. Entering a valid starting reference followed by an invalid entry will also cause the formats from the starting reference to the end of the table to change.

To restore the previous format (and in **OFFLINE** mode, restore the data values stored at the time of the last disk update), press **ALT-A** to abort the change. Pressing **ALT-A** will undo <u>all</u> format changes and value changes made since the last disk update.

To save all the changes and remain in the reference table, press **ALT-U**. To save the changes and exit from the reference table, press the **Escape** key.

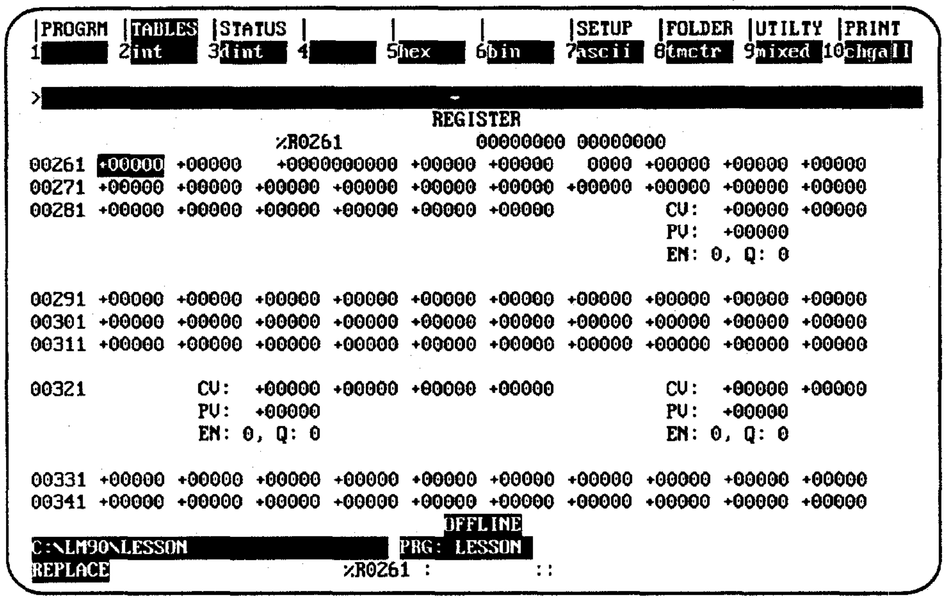
Timer/Counter Format

To support the timer/counter function, a timer/counter format requiring three contiguous reference addresses will display the current value, the preset value, and the control word. In the control word, bit 15 contains the output status (Q), and bit 16 contains the enable status (EN). The rest of the bits in the control word are not used by a counter. For a timer, the rest of the control word bits contain the timer accuracy.

The preset value (PV) can be modified in both **ONLINE** and **OFFLINE** mode. If a reference or constant was assigned to the PV parameter of the timer/counter function, the contents of the reference or the constant are written to the PV reference of the timer/counter format when in **RUN** mode.

Regardless of whether the reference is assigned to a timer or counter, the format can be used as long as the reference type is %R.

The example screen below shows the timer/counter format displayed, using the Timer/Counter (F8) function key.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1       2int    3dint   4       5hex    6bin    7ascii  8tmctr  9mixed 10chgall

>                                    -
                                REGISTER
                  %R0261              00000000 00000000
00261 +00000 +00000   +0000000000 +00000 +00000   0000 +00000 +00000 +00000
00271 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000
00281 +00000 +00000 +00000 +00000 +00000 +00000        CU:    +00000 +00000
                                                       PV:    +00000
                                                       EN: 0, Q: 0

00291 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000
00301 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000
00311 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000

00321        CU:    +00000 +00000 +00000 +00000        CU:    +00000 +00000
             PV:    +00000                             PV:    +00000
             EN: 0, Q: 0                               EN: 0, Q: 0

00331 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000
00341 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000 +00000
                                     OFFLINE
C:\LM90\LESSON                     PRG: LESSON
REPLACE                 %R0261 :           ::
```

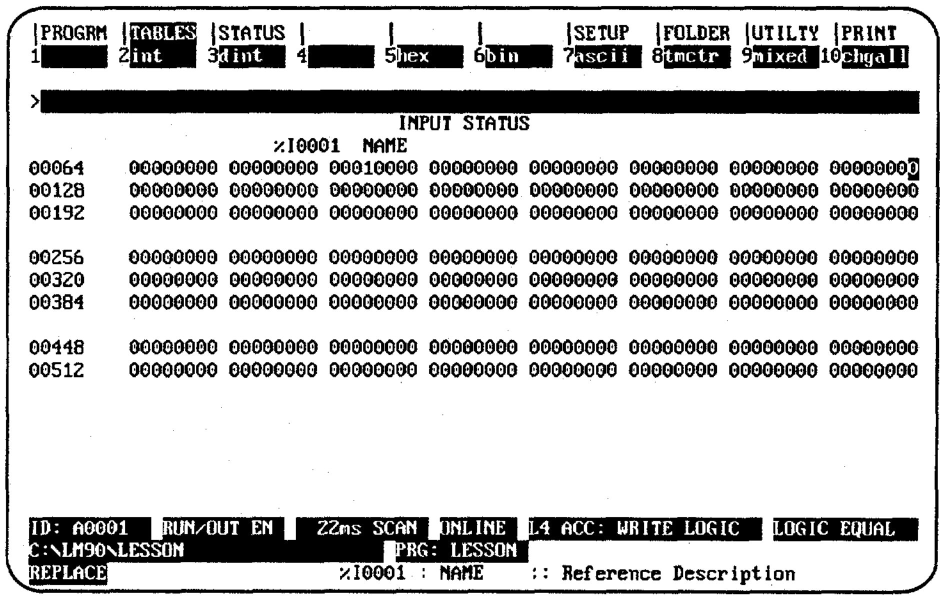
Returning to Default Values

After editing, you can return a standard reference table to its default format and fill the table locations with zeros. This will:

- Set all the reference values in the table to zero.
- Change all references in a discrete reference table back to binary format.
- Change all references in a register reference table back to signed decimal format.

To restore the table to its default content:

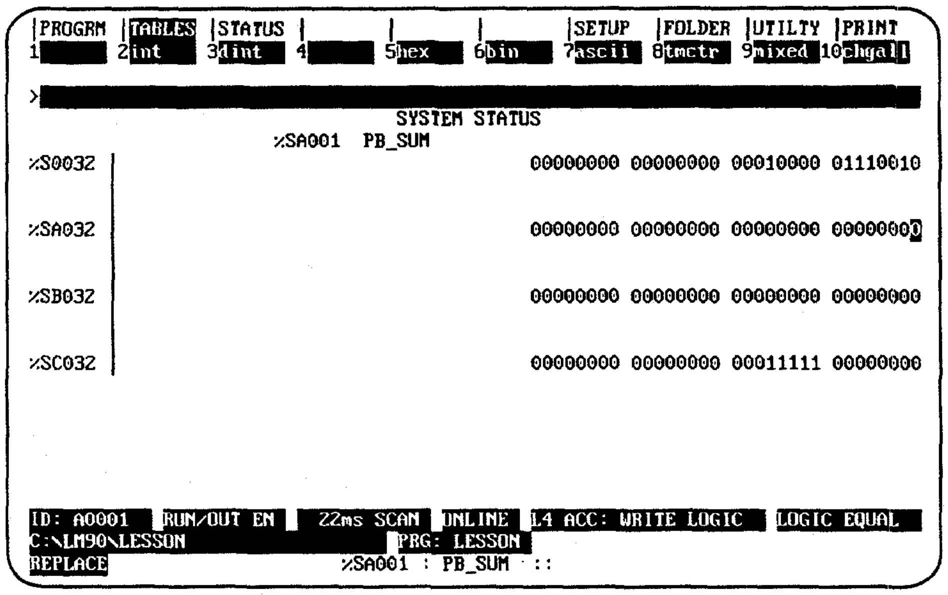
1. Select **Change All (`F10`)**.
2. Enter `0` on the command line, and press the **`Enter`** key. The screen prompts: "Initialize table data and formats to default ? (Y/N".
3. Enter `Y` (Yes) to restore the table's default content. For example:



```
|PROGRM |TABLES |STATUS |        |        |        |SETUP   |FOLDER  |UTILTY  |PRINT
1       2int    3dint   4        5hex     6bin     7ascii   8tmctr   9mixed  10chgall

>
                                  INPUT STATUS
                    %I0001  NAME
00064     00000000 00000000 00010000 00000000 00000000 00000000 00000000 00000000
00128     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00192     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

00256     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00320     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00384     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

00448     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00512     00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000




ID: A0001  RUN/OUT EN   22ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                       PRG: LESSON
REPLACE                      %I0001 : NAME     :: Reference Description
```

System Reference Table

The sample system reference table below is used for system references (%S, %SA, %SB, and %SC), while in **ONLINE** or **MONITOR** mode. (No data is displayed in **OFFLINE** mode.) The system reference table resembles a mixed reference table; however, format changes are not allowed, the Change All (F10) function key cannot be used, and the cursor cannot be moved to blank lines.



```
|PROGRM |TABLES |STATUS |        |        |        |SETUP  |FOLDER |UTILTY |PRINT
1       2int    3dint   4        5hex     6bin     7ascii  8tmctr  9mixed 10chgall
>
                                  SYSTEM STATUS
                        %SA001  PB_SUM
%S0032  |                                   00000000 00000000 00010000 01110010

%SA032  |                                   00000000 00000000 00000000 00000000

%SB032  |                                   00000000 00000000 00000000 00000000

%SC032  |                                   00000000 00000000 00011111 00000000



ID: A0001  RUN/OUT EN   22ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                     PRG: LESSON
REPLACE                    %SA001 : PB_SUM  ::
```

The system reference table is displayed with the cursor on the reference address entered, or the reference address associated with the nickname entered.

In the screen shown above, pressing the **Home** or **Page Up** key will position the cursor on %S0001. Pressing the **End** or **Page Down** key will place the cursor on %SC0032.

Section 5: Mixed Reference Tables

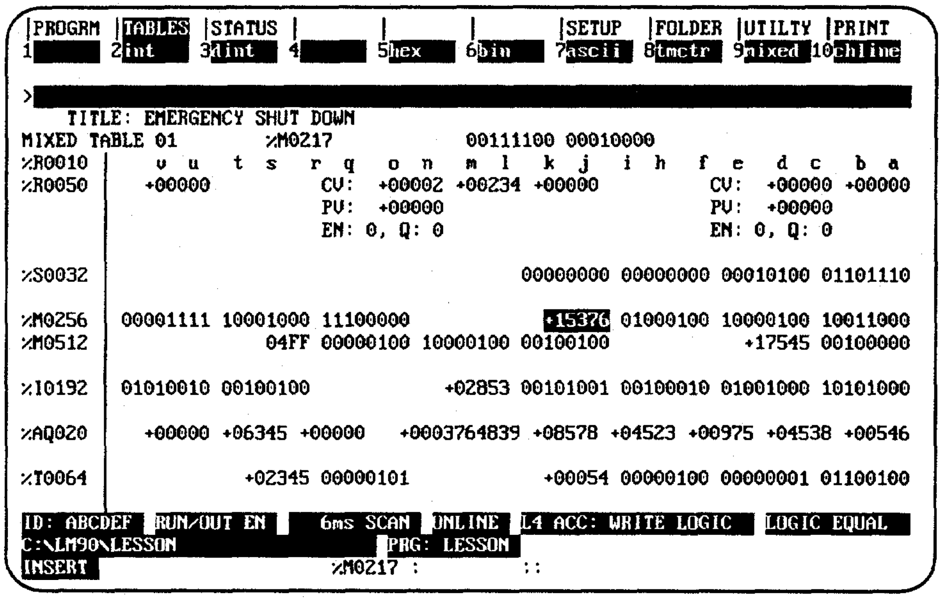
A mixed table represents a collection of data from one or more of the fixed tables. Therefore, the identical information displayed in a mixed table could also be displayed on the related fixed tables.

A total of 99 user-defined mixed reference table displays can be supported. A number from 1 to 99 is associated with each table.

Note

The mixed table function is used only for viewing PLC information; therefore, mixed reference tables are not valid when the programmer mode is **OFFLINE**. When **OFFLINE**, no cursor movement or other mixed table functions are allowed, except for changing tables, exiting reference tables, or changing modes.

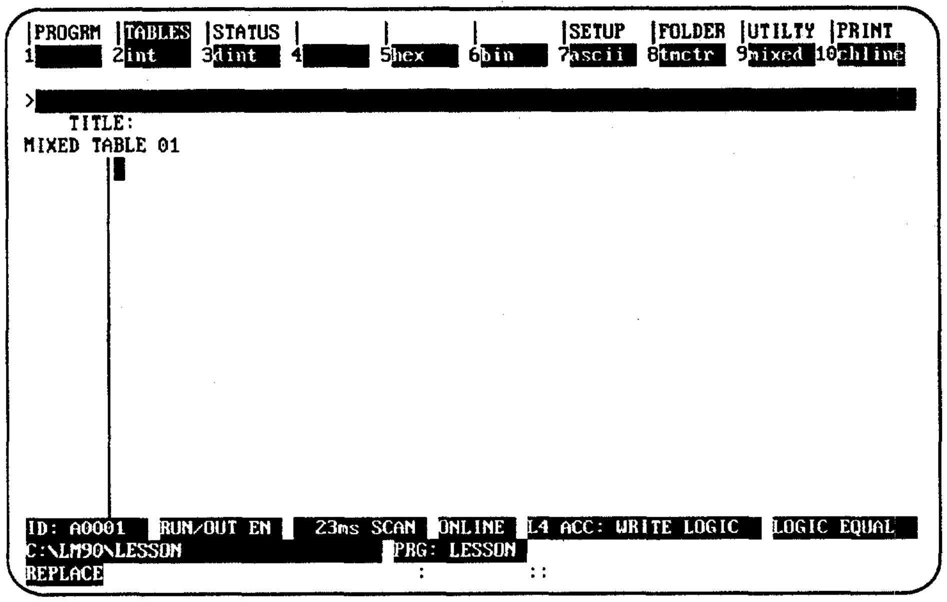
To select a mixed table, enter a number from 1 through 99 on the command line and press **Mixed** (**F9**). This is an example of a mixed reference table display:



```
|PROGRM |TABLES |STATUS |        |        |        |SETUP   |FOLDER  |UTILTY  |PRINT
1       2int    3dint   4        5hex     6bin     7ascii   8tmctr   9mixed   10chline

>
    TITLE: EMERGENCY SHUT DOWN
MIXED TABLE 01          %M0217              00111100 00010000
%R0010 |   v  u    t   s    r  q    o  n    m  l    k  j    i  h    f  e    d  c    b  a
%R0050 |   +00000           CV:  +00002 +00234 +00000               CV:  +00000 +00000
       |                    PU:  +00000                             PU:  +00000
       |                    EN: 0, Q: 0                             EN: 0, Q: 0

%S0032 |                                   00000000 00000000 00010100 01101110

%M0256 | 00001111 10001000 11100000            +15376 01000100 10000100 10011000
%M0512 |                04FF 00000100 10000100 00100100            +17545 00100000

%I0192 | 01010010 00100100             +02853 00101001 00100010 01001000 10101000

%AQ020 |   +00000 +06345 +00000   +0003764839 +08578 +04523 +00975 +04538 +00546

%T0064 |          +02345 00000101             +00054 00000100 00000001 01100100
ID: ABCDEF  RUN/OUT EN    6ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                           PRG: LESSON
INSERT                      %M0217 :              ::
```

Field	Description
User-Defined Table Title	A description of the mixed table. A maximum number of 63 characters is allowed. When the cursor is on the 63rd character, it remains fixed, overwriting the last character.
Mixed Table Number	A number from 1 to 99, which indicates the mixed table currently displayed.
Cursor Item Line	Information about the address the cursor is on; namely, the current reference, reference nickname, and current value binary representation.
Table Data	16 display lines. Some may be program block headers or blank lines.

If the mixed table entered is not defined, a blank mixed table (no data, no title, only the mixed table number) is displayed, as shown below.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1       2int    3dint   4       5hex    6bin    7ascii  8tmctr  9mixed 10chline
>
    TITLE:
MIXED TABLE 01

ID: A0001  RUN/OUT EN   23ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                     PRG: LESSON
REPLACE                             :          : :
```

Editing the Title

To edit the table title, position the cursor on the title of the display by moving the cursor to the topmost data display line and pressing the **Up** cursor key. While editing the title, the Update or Abort key can be used to save or restore a title. A message is displayed when the function is complete.

Note

Since mixed reference tables are defined on a line-by-line basis, the abort function will restore the lines, formats, and title displayed upon entry. Unless an update to disk was done, any lines defined or title edited are lost.

The **Mixed** (**F9**) key is used to change from one mixed table to another. The **Tables** (**Shift-F2**) key sequence can also be used to get to any table. The **Change All** (**F10**) key is not supported for global format changes, as in fixed table displays.

To return to the display area, press the **Enter** key or the **Down** cursor key. The cursor will return to the original position in the topmost data display line.

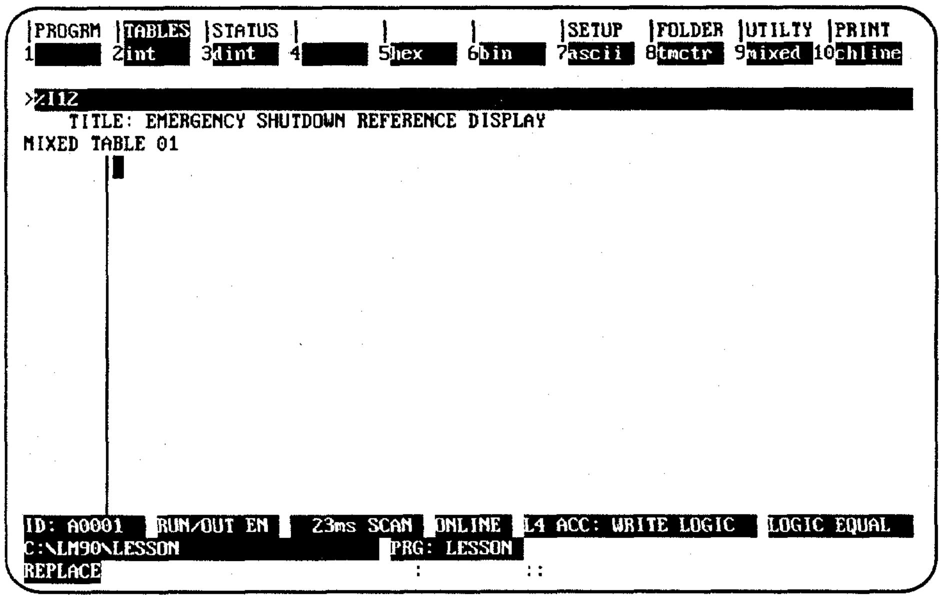
Defining a Mixed Table

A mixed reference table display is created or changed by pressing the **Change Line** (**F10**) key.

1. First, position the cursor on the display line where a certain data value should be displayed. The line may already be defined, or it may be a blank line.

2. Enter the reference address or nickname of the desired data value on the command line.

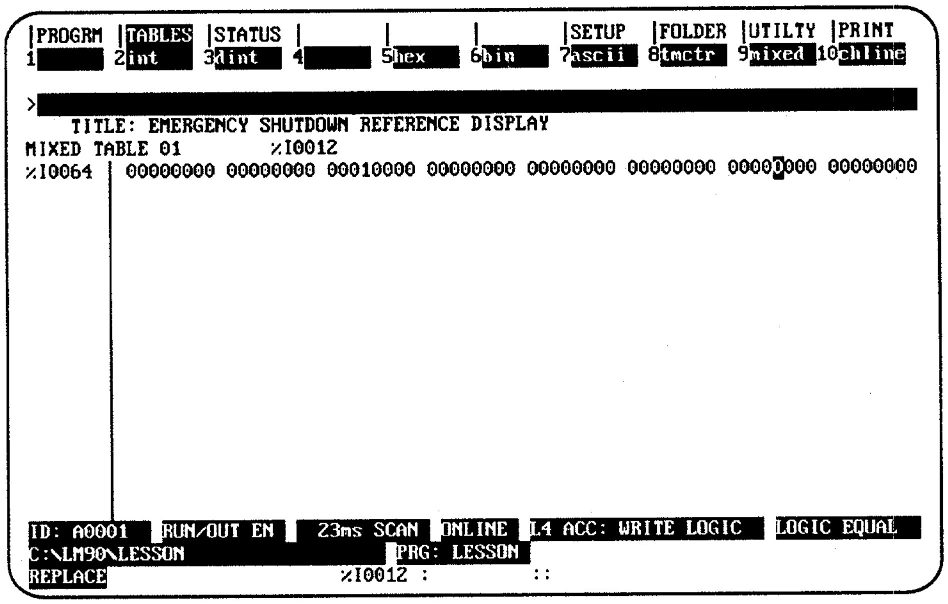
   In the following example, a line is defined in mixed reference table 1. With the cursor positioned on the top line of the display, enter **%I1** onto the command line.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1       2int    3dint   4       5hex    6bin    7ascii  8tmctr  9mixed  10chline
>%I1Z
    TITLE: EMERGENCY SHUTDOWN REFERENCE DISPLAY
MIXED TABLE 01

ID: A0001   RUN/OUT EN    23ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                          PRG: LESSON
REPLACE                                  :            ::
```

3. Press **Change Line** (**F10**) to display the desired information. The cursor will be positioned on the requested data value. Any information currently displayed is replaced with the new requested information.

   In this example, when **F10** is pressed, the data value positioned at reference %I0012 is entered into the topmost display line.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1       2int    3dint   4       5hex    6bin    7ascii  8tmctr  9mixed  10chline

>
   TITLE: EMERGENCY SHUTDOWN REFERENCE DISPLAY
MIXED TABLE 01        %I0012
%I0064 | 00000000 00000000 00010000 00000000 00000000 00000000 00000000 00000000

ID: A0001  RUN/OUT EN  23ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                     PRG: LESSON
REPLACE                        %I0012 :          ::
```

The default display format for data extracted from a discrete table is binary; for register tables, it is signed integer.

The timer/counter format can be used for a %R line in a mixed table if enough exists for the format. If the two lines below the %R line are defined or are beyond the table size, an error message is displayed.

Deleting a Line

If the command line is blank when the **Change Line** (**F10**) key is pressed, the line is cleared and the cursor is automatically moved to the next line.

If the current line is already blank, moving the cursor to the next line will produce the same result as pressing **F10** with the command line blank.

Moving the Cursor

Cursor movement is the same in mixed tables as in fixed tables. Since blank lines can be defined in a mixed table, the cursor is allowed on blank lines.

Cursoring up is allowed when the cursor is on the topmost line, in order to access the title. You can cursor to lines which contain a message, indicating that the reference is out of range. This allows you to delete or re-define the line.

The same line of references can also be defined twice on the same mixed table display. If the Enter key is used to move the cursor to a duplicate reference address, the search for the reference address begins with the next reference and continues to the end of the table. If the reference address is not found in that section, the search begins with the first non-blank line, lowest reference. If the reference address is still not found, an error message is displayed.

The Enter key cannot be used to change tables.

Timer/Counter Format

The timer/counter format in a mixed table is the same as in a fixed table, except for the first timer/counter format of the line. In the timer/counter format, the next two lines must be blank so that there is sufficient room for the format. Otherwise, an error message is displayed.

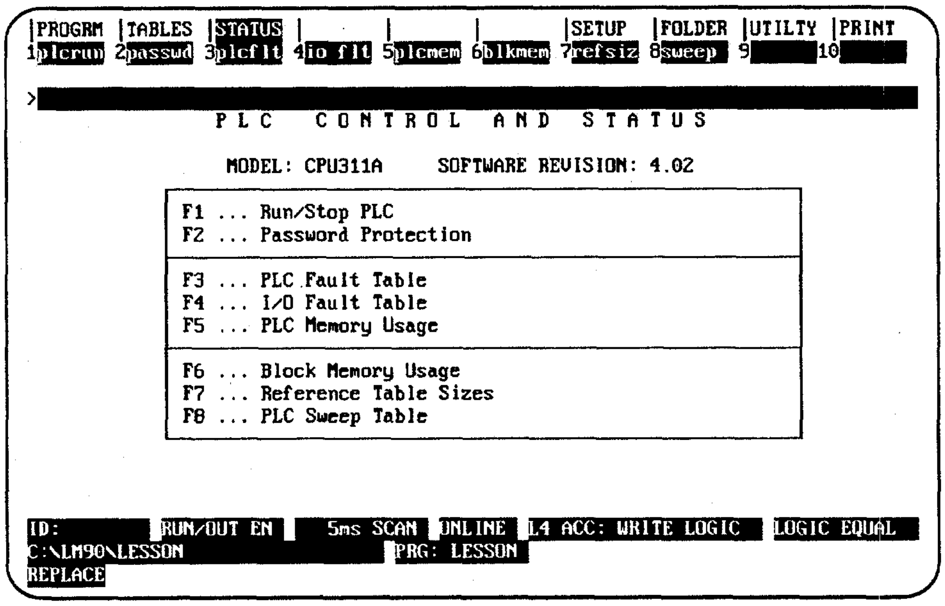
Chapter 5

PLC Control and Status

The programmer can interact with an operating PLC in many ways. Monitoring program execution and reference tables was described in previous chapters. Several additional features involve interaction between the programmer and an operating PLC. To use most of them, the programmer must be connected to the PLC and must be in **ONLINE** or **MONITOR** mode.

If the programmer is not connected or is in **OFFLINE** mode, asterisks may be displayed in place of values in many fields when the function screens are displayed. In addition, some of these features may be protected by passwords.

To use these features, press **STATUS** (**F3**) from the main menu, or **Shift-F3** from any other main menu function.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1plcrun 2passwd 3plcflt 4io flt 5plcmem 6blkmem 7refsiz 8sweep  9       10
>
                    P L C   C O N T R O L   A N D   S T A T U S

                     MODEL: CPU311A    SOFTWARE REVISION: 4.02

              F1 ... Run/Stop PLC
              F2 ... Password Protection

              F3 ... PLC Fault Table
              F4 ... I/O Fault Table
              F5 ... PLC Memory Usage

              F6 ... Block Memory Usage
              F7 ... Reference Table Sizes
              F8 ... PLC Sweep Table

ID:          RUN/OUT EN     5ms SCAN ONLINE L4 ACC: WRITE LOGIC  LOGIC EQUAL
C:\LM90\LESSON                      PRG: LESSON
REPLACE
```

Note

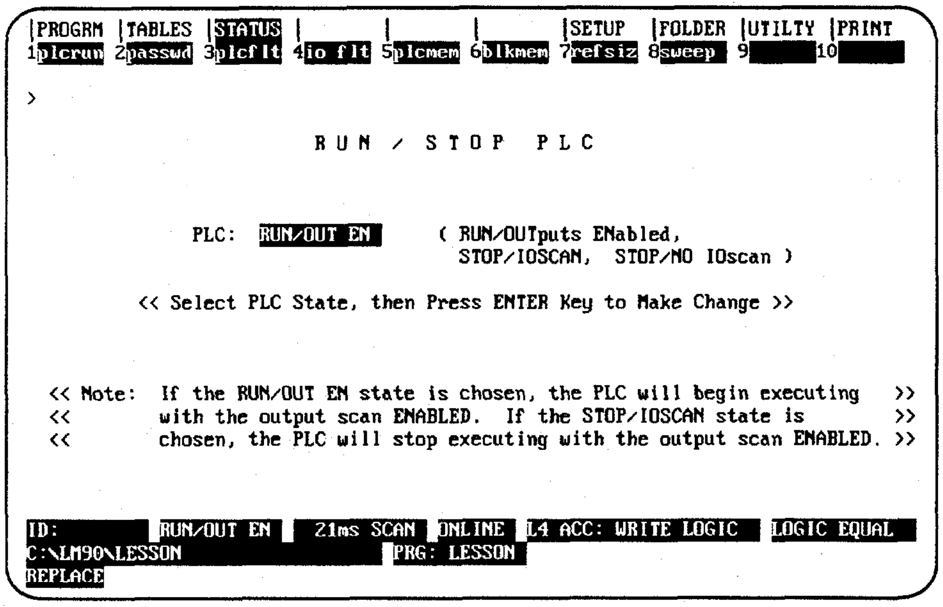
The screen displayed above identifies the CPU model which is attached and its software revision.

The **F9** and **F10** softkeys are now blank when the PLC Control and Status menu is displayed. **F9**, which had previously been designated as the OEM softkey, is not present unless the active function is OEM protection. When the function is OEM protection, F9 will become the OEM softkey.

F10, which had previously been designated as the Clear softkey, is not used unless the active function is either the PLC fault table or the I/O fault table. When either fault table function is active, **F10** will become the **Zoom** softkey and **F9** will become the **Clear** softkey.

Run/Stop the PLC

To start or stop program execution in the PLC, and to determine the state of outputs if the PLC is started, press **PLC Run** (**F1**) from the PLC Control and Status menu or from another PLC functions screen. Communications between the programmer and the PLC must already be established.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1plcrun 2passwd 3plcflt 4io flt 5plcmem 6blkmem 7refsiz 8sweep  9       10

>

                         R U N  /  S T O P   P L C


                  PLC:  RUN/OUT EN     ( RUN/OUTputs ENabled,
                                         STOP/IOSCAN,  STOP/NO IOscan )

              << Select PLC State, then Press ENTER Key to Make Change >>


 << Note:  If the RUN/OUT EN state is chosen, the PLC will begin executing   >>
 <<        with the output scan ENABLED.  If the STOP/IOSCAN state is        >>
 <<        chosen, the PLC will stop executing with the output scan ENABLED. >>


ID:          RUN/OUT EN   21ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                        PRG: LESSON
REPLACE
```

If the programmer is in **OFFLINE** mode, default values are displayed on the screen. Press **ALT-M** to change the programmer operating mode to **ONLINE** or **MONITOR** mode. (For the Workmaster and CIMSTAR I computers, use the keyswitch. Refer to appendix C, "Programmer Environment Setup," for instructions on using the keyswitch.) **ALT-R** can also be used to toggle the PLC mode between **RUN** and the configured **STOP** mode.

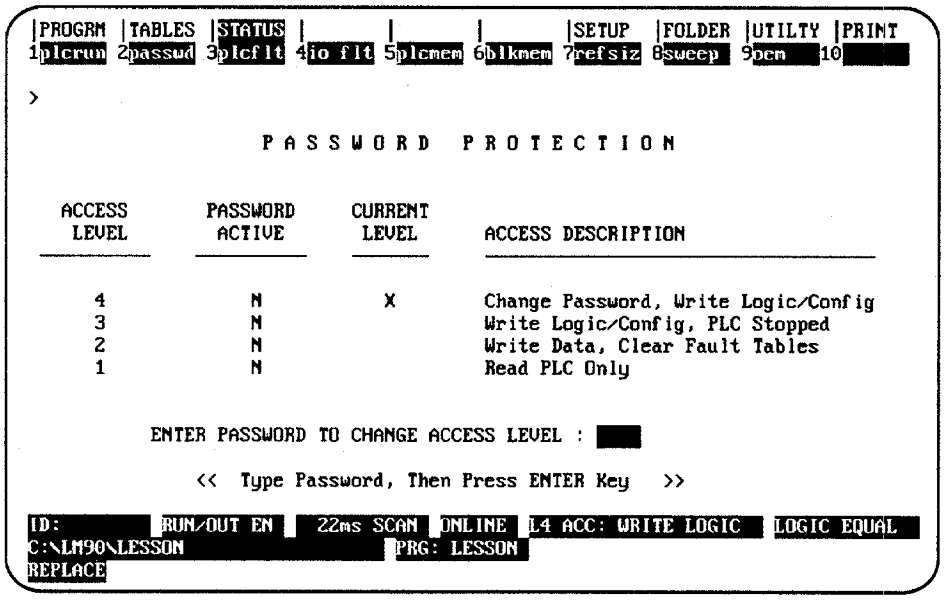
The *PLC* field on the screen shown above indicates whether the attached PLC is to be started or stopped. When the programming software is initially executed, this field is initialized to **STOP**. Use the **Tab** key to select one of the modes listed below; then, press the **Enter** key.

Mode	Description
Run/Outputs Enabled	The PLC is running the logic program with outputs enabled.
Stop/IOScan	The PLC is stopped, not executing the logic program, but is scanning I/O.
Stop/No IOScan	The PLC is stopped, not executing the logic program, no I/O scan.

PLC Password Protection

PLC password protection can be used to restrict access to selected PLC functions. After passwords have been set up, a protected function cannot be used unless the proper password has been entered. Password protection is not intended to restrict access to the programming or configuration software. If PLC communications are suspended, protection level will automatically return to the highest unprotected level.

To display or change the current protection level of the PLC, the programmer must be in **ONLINE** or **MONITOR** mode and communicating with the intended PLC. Select **Password** (**F2**) from the PLC Control and Status menu or from another PLC functions screen.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1plcrun 2passwd 3plcflt 4io flt 5plcmem 6blkmem 7refsiz 8sweep  9pcm    10

>

                      P A S S W O R D   P R O T E C T I O N

   ACCESS      PASSWORD      CURRENT
    LEVEL        ACTIVE        LEVEL      ACCESS DESCRIPTION
  ----------   ----------   -------    ------------------------------------

      4            N            X        Change Password, Write Logic/Config
      3            N                     Write Logic/Config, PLC Stopped
      2            N                     Write Data, Clear Fault Tables
      1            N                     Read PLC Only

           ENTER PASSWORD TO CHANGE ACCESS LEVEL :

               <<  Type Password, Then Press ENTER Key  >>

ID:           RUN/OUT EN   22ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                       PRG: LESSON
REPLACE
```

If the screen appears without entries, the programmer is in **OFFLINE** mode. Press **ALT-M** to change the programmer operating mode to **ONLINE** or **MONITOR** mode. (For the Workmaster and CIMSTAR I computers, use the keyswitch.)

Field	Description
Access Level	Each access level includes all the privileges of lower levels: • <u>Level 4</u>: Write to all configuration or logic. Configuration may only be written in **STOP** mode; logic may be written in **STOP** or **RUN** mode. Display, set, or delete passwords for any level. (This is the default if no passwords are assigned.) • <u>Level 3</u>: Write to any configuration or logic, including word-for-word changes, the addition/deletion of program logic, and the overriding of discrete I/O. • <u>Level 2</u>: Write to any data memory, except overriding discrete I/O. The PLC can be started or stopped. PLC and I/O fault tables can be cleared. • <u>Level 1</u>: Read any PLC data except passwords; no PLC memory may be changed.
Password Active	A **Y** (Yes) displayed in this column indicates that a password has been assigned to this access level in the PLC.
Current Level	An **X** displayed in this column indicates the current protection level of the PLC.
Access Description	Access allowed at each level. Privileges accumulate as the level increases; at any given level, all privileges at lower levels are permitted.

Changing the Privilege Level

To access any level, the programmer must be in **ONLINE** or **MONITOR** mode and communicating with the PLC.

1. Move the cursor to the line at the bottom of the screen which prompts you to enter a password to change the access level.
2. Type a password, consisting of up to 4 ASCII hexadecimal digits (0 – 9, A – F), and press the **Enter** key. Characters are not displayed as you type them.

A valid password enables you to access the protection level for that password, and also all protection levels below that level.

Creating, Changing, or Removing Passwords

Passwords are created in the configuration software, using the status function as described below. They are then transmitted to the PLC, where they are stored.

Caution

The PLC does not save passwords through a loss of power, unless the CPU battery is attached. After restarting the PLC, passwords must be sent to the PLC again.

The use of passwords is an optional feature; it is not necessary to use any passwords at all. You can also use passwords to restrict access to some PLC features but not to others. Note, however, that the PLC always defaults to the highest unprotected level, so there should not be any "gaps" in protection levels.

To enter or change passwords, the computer must be in **ONLINE** mode and communicating with the PLC. Entering or changing passwords requires access to the highest level. If no passwords have been set up for the system, this level is automatically available. Once passwords have been entered, they can only be changed by performing one of the following tasks:

- Enter the correct password to access the highest-level privileges.

OR

- In the configuration software, by place the master diskette No. 1 in the system disk drive of the computer and press the **ALT** and **O** keys. Because this allows passwords to be overridden or changed without entering the correct password, it is important to keep the original software master diskettes in a secure location.

To create, change, or remove passwords:

1. Access the Password Protection screen in the configuration software by pressing **Password** (**F2**) from the PLC Control and Status menu. A column titled "Password" appears on this screen; this column is not displayed in the programming software. Use this column to enter new passwords.
2. If there is not already a password set for the highest level, begin by creating one. This will protect the passwords you enter against unauthorized changes.
3. In the password column, locate the cursor at a level you want to protect or unprotect. Then:

   A. To create or change a password, enter a password consisting of up to 4 ASCII hexadecimal digits (0 - 9, A - F).

   B. To remove an existing password previously stored to the PLC, enter four blank characters or press **ALT-C** to clear the field.

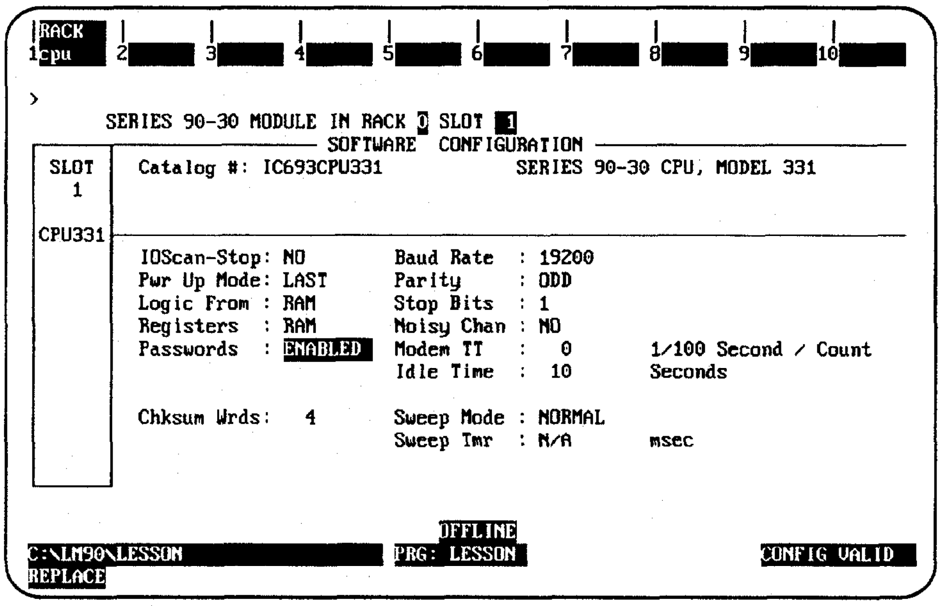
   C. Press the **Enter** key to validate your entry.
4. To save each new password to the PLC, press the **Enter** key again. Respond to the prompt that appears by pressing the **Y** (Yes) key.

   If you want to quit without saving any passwords to the PLC, press the **N** (No) key.
5. To exit, use the **Escape** key or any main menu function key.

Enabling/Disabling Passwords

The ability to use passwords may be enabled or disabled in the configuration software. You may want to disable this feature to prevent someone from setting passwords in the CPU.

A sample CPU module detail screen displaying the *Passwords* field that would be used to enable or disable the password feature is shown below. To access this screen, press **Zoom** (**F10**) with the cursor positioned on the CPU-configured slot on the Rack Configuration screen.



```
|RACK   |      |      |      |      |      |      |      |      |
1cpu    2      3      4      5      6      7      8      9      10
>
      SERIES 90-30 MODULE IN RACK 0 SLOT 1
 ____________________________ SOFTWARE  CONFIGURATION ____________________
| SLOT | Catalog #: IC693CPU331            SERIES 90-30 CPU, MODEL 331
|  1   |
|CPU331|__________________________________________________________________
|      |  IOScan-Stop: NO         Baud Rate   : 19200
|      |  Pwr Up Mode: LAST       Parity      : ODD
|      |  Logic From : RAM        Stop Bits   : 1
|      |  Registers  : RAM        Noisy Chan  : NO
|      |  Passwords  : ENABLED    Modem TT    :   0      1/100 Second / Count
|      |                          Idle Time   :  10      Seconds
|      |
|      |  Chksum Wrds:    4       Sweep Mode  : NORMAL
|      |                          Sweep Tmr   : N/A      msec
|______|

                                 OFFLINE
C:\LM90\LESSON                   PRG: LESSON                  CONFIG VALID
REPLACE
```

The values for the *Password* field are **ENABLED** (default selection) or **DISABLED**. Entering **ENABLED** will allow the password feature to be used; entering **DISABLED** will prevent the password feature from being used.

Caution

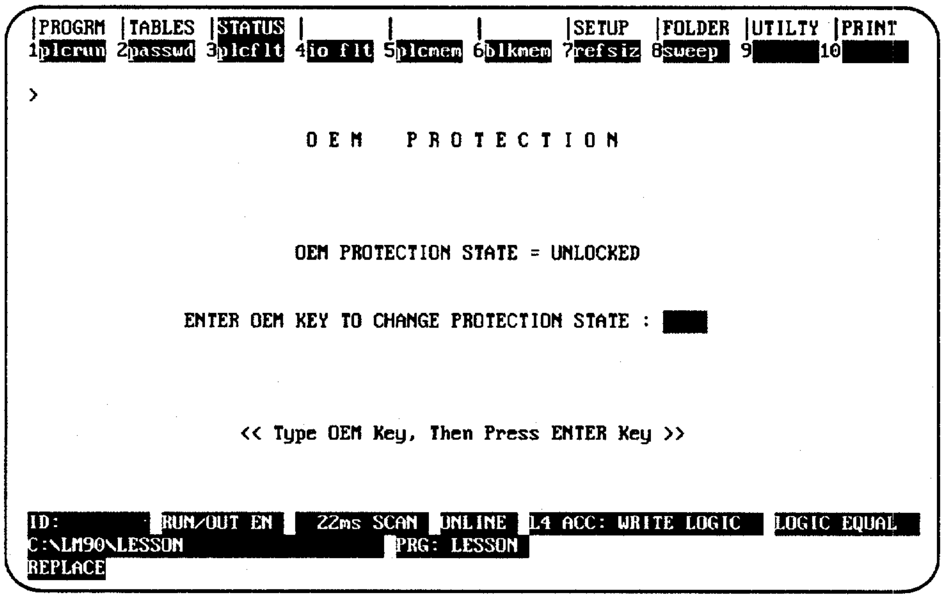
In order to re-enable passwords once they have been disabled, PLC memory must be cleared with an HHP. The HHP needs to be connected. Then power off the PLC. Then hold both the <CLR> and <M/T> keys down while powering the PLC back up. If you do not have an HHP, call the GE Fanuc Technical Service Hotline (1-800-828-5747) for assistance.

Please refer to chapter 10, section 3, "Configuring the CPU Module," for more information on configuring this module.

OEM Protection

The OEM protection feature provides a higher level of security than password protection. It may be used to further restrict access to program logic and configuration parameters.

To display the OEM Protection screen, press **OEM** (**F9**) from the Password Protection screen.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1plcrun 2passwd 3plcflt 4io flt 5plcmem 6blkmem 7refsiz 8sweep  9       10

>
                          O E M    P R O T E C T I O N


                        OEM PROTECTION STATE = UNLOCKED

                 ENTER OEM KEY TO CHANGE PROTECTION STATE :


                     << Type OEM Key, Then Press ENTER Key >>

ID:          RUN/OUT EN   22ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                      PRG: LESSON
REPLACE
```

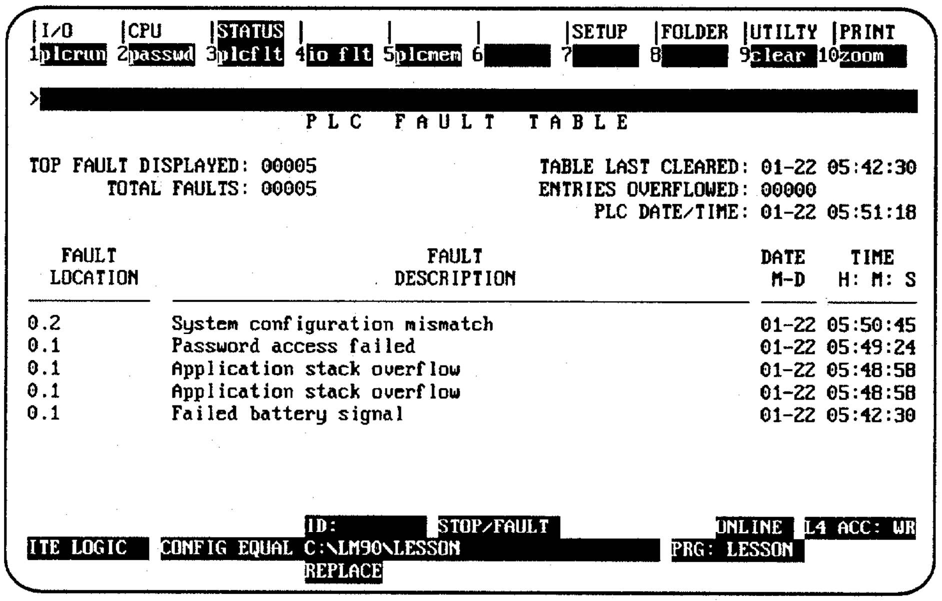
OEM protection is enabled and disabled by entering the OEM key. The OEM key is a 1 to 4 hexadecimal digit password. If the OEM key has never been set, it will equal the NULL (blank) string. Definition of the OEM key may only be performed in the configuration software; however, OEM protection may be locked or unlocked in the programming software.

The OEM protection state is toggled between unlocked and locked each time the OEM key is correctly entered. The OEM protection state cannot be toggled to locked until the OEM key is set to something other than the NULL (blank) string. Once the OEM protection is locked, it will remain locked until the OEM key is correctly entered to toggle the protection state.

The OEM protection state is retentive across a power cycle. If the OEM protection is locked in the PLC and power is lost, the OEM protection state will remain locked when power is restored to the PLC if the CPU battery is attached.

PLC Fault Table

The PLC Fault Table screen lists PLC faults such as password violations, PLC/configuration mismatches, parity errors, and communications errors. For example:



```
|I/O     |CPU     |STATUS |       |        |        |SETUP   |FOLDER |UTILTY |PRINT
1plcrun  2passwd  3plcflt  4io flt 5plcmem  6        7        8       9clear  10zoom
>
                         P L C   F A U L T   T A B L E

TOP FAULT DISPLAYED: 00005                      TABLE LAST CLEARED: 01-22 05:42:30
      TOTAL FAULTS: 00005                       ENTRIES OVERFLOWED: 00000
                                                    PLC DATE/TIME: 01-22 05:51:18

   FAULT                        FAULT                               DATE    TIME
  LOCATION                    DESCRIPTION                            M-D   H: M: S

0.2          System configuration mismatch                          01-22 05:50:45
0.1          Password access failed                                 01-22 05:49:24
0.1          Application stack overflow                             01-22 05:48:58
0.1          Application stack overflow                             01-22 05:48:58
0.1          Failed battery signal                                  01-22 05:42:30

                        ID:         STOP/FAULT                ONLINE   L4 ACC: WR
ITE LOGIC   CONFIG EQUAL C:\LM90\LESSON                 PRG: LESSON
                        REPLACE
```

To display the PLC Fault Table screen, press **PLC Fault** (**F3**) from the PLC Control and Status menu or from another PLC functions screen. The programmer may be in any operating mode. However, if the programmer is in **OFFLINE** mode, no faults are displayed. In **ONLINE** or **MONITOR** mode, PLC fault data is displayed. In **ONLINE** mode, faults can be cleared (this may be password protected).

Field	Description
Top Fault Displayed	The index of the PLC fault currently at the top of the fault display is shown on the first line of this screen.
Total Faults	The total number of faults since the table was last cleared.
Table Last Cleared	The date and time faults were last cleared from the fault table. This information is maintained by the PLC.
Entries Overflowed	The number of entries lost because the fault table has overflowed since it was cleared. The PLC fault table can contain up to 16 faults.
PLC Time/Date	The current date and time. This is also maintained by the PLC.

Note

Because the Model 323, 321, 311, 313, 211, and Micro CPUs do not support the time-of-day clock, entries for date and time in the fault tables are displayed as 00-00 00:00:00.

Fault Table Entries

For each fault, the display shows the date and time the fault occurred, and the following information about the fault:

Field	Description
Fault Location	The location of the fault (rack/slot address). For example, 3.2 refers to rack 3, slot 2.
Fault Description	The fault that has occurred: Loss of, or missing, I/O module. Loss of, or missing, option module. Addition of, or extra, rack. Addition of, or extra, I/O module. System configuration mismatch. PLC CPU hardware failure. [1] Non-fatal module hardware failure. Option module software failure. Program checksum failure. Low battery signal. Constant sweep time exceeded. PLC system fault table full. User application fault. No user program present. [2] Corrupted user program on power-up. Window completion failure. Password access failure. Null system configuration for RUN. [3] PLC CPU system software failure. [4] Communications failure during store. [5]

1 Running the system with null system configuration is the same as having I/O scans suspended.
2 If this fault occurs, the PLC stops all activity.
To clear this fault, you must cycle power to the PLC.
3 If this fault is present, the PLC will not transition to **RUN** mode. This fault must be cleared from the programmer (not cleared by cycling power).

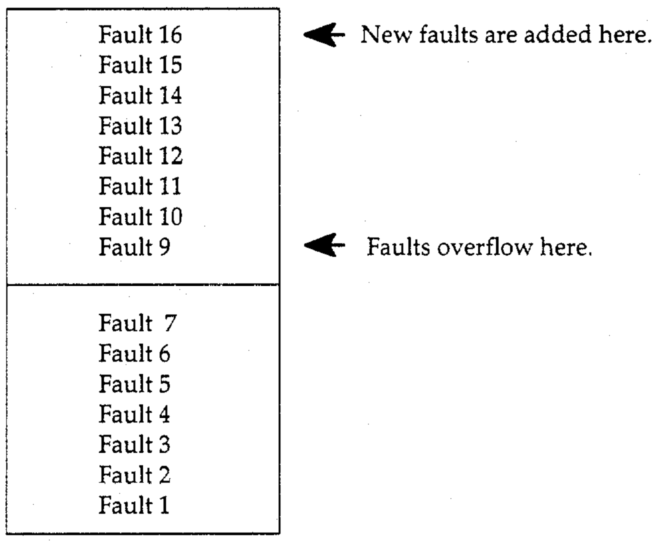
For diagnostic faults, the CPU sets fault references. For fatal faults, the CPU sets fault references and places the CPU in **STOP** mode.

If there are more faults than will fit on one screen, you can display them using the Home, End, Page Up and Page Down, and cursor keys. Refer to the *Series 90-30/20/Micro Programmable Controllers Reference Manual,* GFK-0467, for more fault information.

Number of Faults in the PLC Fault Table

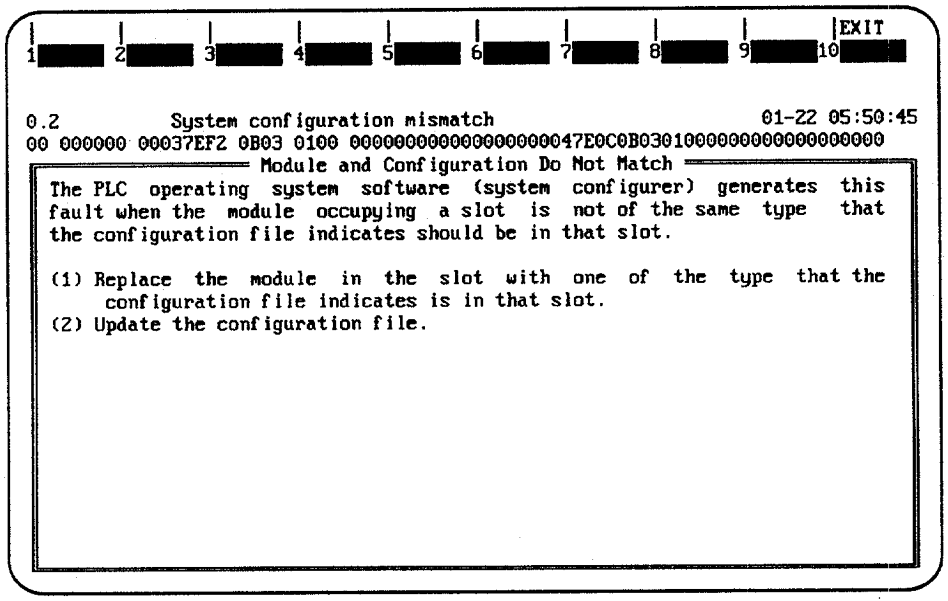
The PLC fault table can contain up to 16 faults. Additional faults cause the table to overflow, and faults are lost. The system reference SY_FULL (%S0009) is set to indicate that the fault table is full.

As faults occur, the first 8 faults are logged into the table and remain there until the table is cleared. None of these eight faults is dropped if the table overflows. For faults 9 through 16, however, the fault table operates as a First-In-First-Out queue. When fault 17 occurs, fault 9 is dropped from the table. Clearing the fault table removes all the fault entries.



Zooming into the PLC Fault Table

The **Zoom (F10)** softkey enables you to obtain additional information pertaining to each fault listed in the PLC fault table. By moving the cursor to a particular fault and pressing **Zoom (F10)**, information about the error code, default action, description of the error, and appropriate corrective action is displayed on a screen similar to this one for a low battery signal.



```
|         |         |         |         |         |         |         |         |         |EXIT
1         2         3         4         5         6         7         8         9         10

0.2          System configuration mismatch                                   01-22 05:50:45
00 000000 00037EF2 0B03 0100 00000000000000000000047E0C0B030100000000000000000
═══════════════════ Module and Configuration Do Not Match ═══════════════════
 The PLC  operating  system  software  (system  configurer)  generates  this
 fault when the  module  occupying  a slot  is  not of the same  type  that
 the configuration file indicates should be in that slot.

 (1) Replace  the  module  in  the  slot  with  one  of  the  type  that the
      configuration file indicates is in that slot.
 (2) Update the configuration file.
```

Note

A hexadecimal dump of the fault can be displayed by pressing **CTRL-F** from this screen. For more information on fault explanations and correction and **CTRL-F**, please refer to the *Series 90-30/20/Micro Programmable Controllers Reference Manual,* GFK-0467.

All softkeys are disabled on this screen, except for **Exit** (**Shift-F10**). The **Abort** key (**ALT-A**) and the **Escape** key may also be used to exit from this screen and return to the PLC fault table.

Clearing the Fault Table

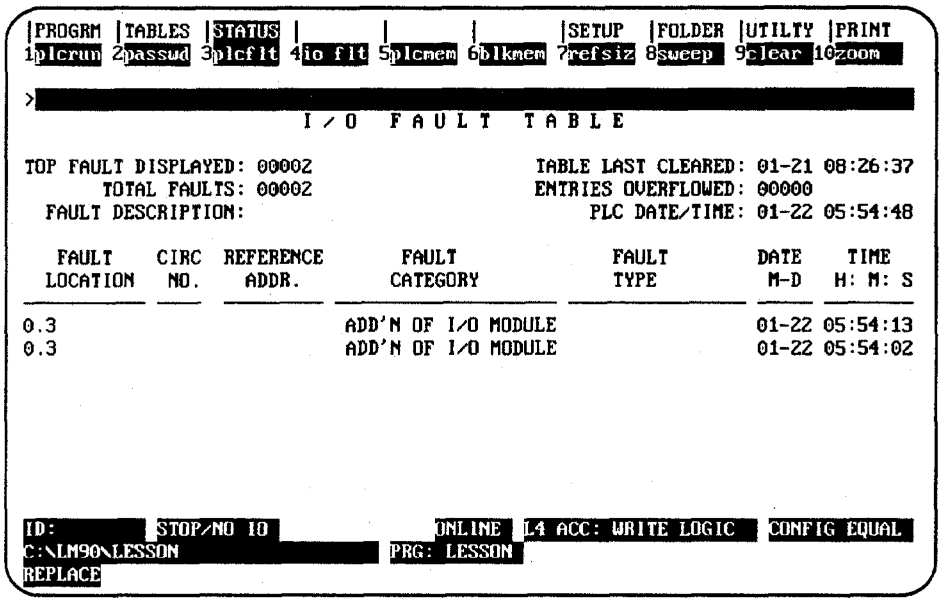
Pressing **Clear** (**F9**) in **ONLINE** mode clears all faults from the fault table. Faults can also be cleared from the program logic using the SVCREQ function, as described in the *Series 90-30/20/Micro Programmable Controllers Reference Manual,* GFK-0467. If passwords have been enabled in the PLC, you must be at level 2 or higher in order to clear the fault table.

If a printer is connected to the programmer, you may want to print a copy of the fault table screen by pressing the **Print Screen** key before clearing the fault table.

Clearing the fault table, of course, does not clear fault conditions in the system. If the condition that caused a fault still exists, the fault may be reported again after storing the configuration, cycling power to the PLC, or during a **STOP-TO-RUN** transition.

I/O Fault Table

The I/O Fault Table screen lists I/O faults such as circuit faults, address conflicts, forced circuits, and I/O bus faults. For example:



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1plcrun 2passwd 3plcflt 4io flt 5plcmem 6blkmem 7refsiz 8sweep  9clear  10zoom
>
                              I / O   F A U L T   T A B L E

TOP FAULT DISPLAYED: 00002                        TABLE LAST CLEARED: 01-21 08:26:37
      TOTAL FAULTS: 00002                         ENTRIES OVERFLOWED: 00000
  FAULT DESCRIPTION:                                  PLC DATE/TIME: 01-22 05:54:48

   FAULT     CIRC  REFERENCE       FAULT                  FAULT        DATE    TIME
  LOCATION   NO.     ADDR.        CATEGORY                TYPE          M-D   H: M: S

0.3                          ADD'N OF I/O MODULE                       01-22 05:54:13
0.3                          ADD'N OF I/O MODULE                       01-22 05:54:02

ID:          STOP/NO IO                   ONLINE  L4 ACC: WRITE LOGIC   CONFIG EQUAL
C:\LM90\LESSON                            PRG: LESSON
REPLACE
```

To display the I/O Fault Table screen, press **I/O Fault** (**F4**) from the PLC Control and Status menu or from another PLC functions screen. The programmer may be in any operating mode. However, if the programmer is in **OFFLINE** mode, no faults are displayed. In **ONLINE** or **MONITOR** mode, PLC fault data is displayed. In **ONLINE** mode, faults can be cleared (this feature may be password protected).

Field	Description
Top Fault Displayed	The index of the I/O fault currently at the top of the fault display is shown on the first line of this screen.
Total Faults	The total number of faults since the table was last cleared.
Fault Description	An explanation of the fault that is currently highlighted in the I/O fault table.
Table Last Cleared	The date and time faults were last cleared from the fault table. This information is maintained by the PLC.
Entries Overflowed	The number of entries lost because the fault table has overflowed since it was cleared. The I/O fault table can contain up to 32 faults.
PLC Time/Date	The current date and time. This is also maintained by the PLC.

Note

Because the Model 311, 313, 211 and Micro CPUs do not support the time-of-day clock, entries for date and time in the fault tables are displayed as 00-00 00:00:00.

Fault Table Entries

For each fault, the display shows the date and time the fault occurred, and the following information about the fault:

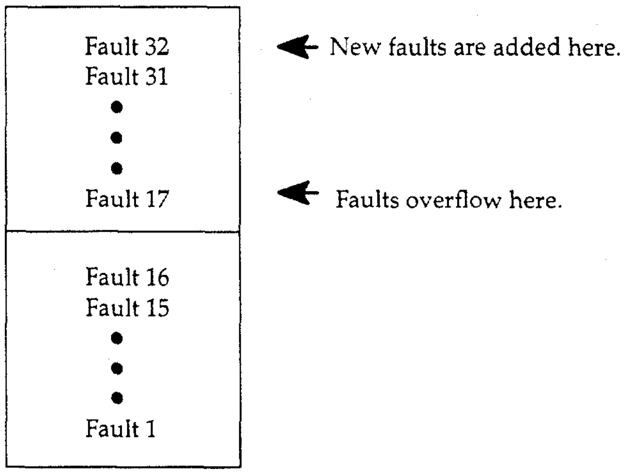
Field	Description
Fault Location	The location of the fault (rack/slot address). For example, 3.2 refers to rack 3, slot 2.
Circuit Number	The relative position of a point within its module. The value may be from 0 to 1023.
ReferenceAddress	The I/O reference address where the fault was detected. The address consists of a two or three character identifier (%I, %Q, %IQ, %AI, %AQ) specifying the memory type and a five-digit offset within the memory type.
Fault Category	The general type of fault that has occurred. For more information, refer to GFK-0467. For diagnostic faults, the CPU sets fault references. For fatal faults, the CPU sets fault references and places the CPU in **STOP** mode.
Fault Type	This further explains the fault category. For more information, refer to GFK-0467.

If there are more faults than will fit on one screen, you can display them using the Home, End, Page Up and Page Down, and cursor keys.

Number of Faults in the I/O Fault Table

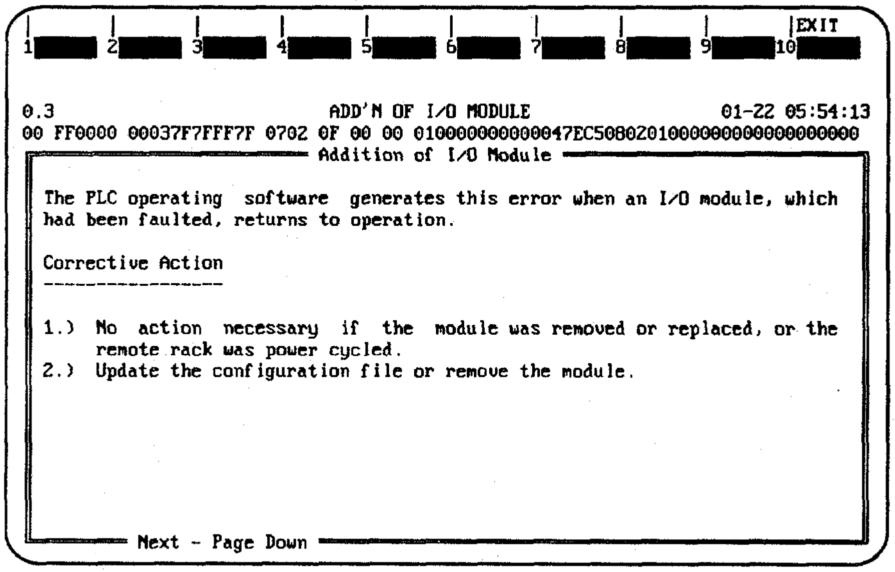
The I/O fault table can contain up to 32 faults. Additional faults cause the table to overflow, and faults are lost. The system reference IO_FULL (%S0010) is set to indicate that the fault table is full.

As faults occur, the first 16 faults are logged into the table and remain there until the table is cleared. None of these 16 faults is dropped if the table overflows. For faults 17 through 32, however, the fault table operates as a First-In-First-Out queue. When fault 33 occurs, fault 17 is dropped from the table. Clearing the fault table removes all the fault entries.



Zooming into the I/O Fault Table

The **Zoom (F10)** softkey enables you to obtain additional information pertaining to each fault listed in the I/O fault table. By moving the cursor to a particular fault and pressing **Zoom (F10)**, information about the error code, default action, description of the error, and appropriate corrective action is displayed on a screen similar to this one for an Addition of I/O Module error.

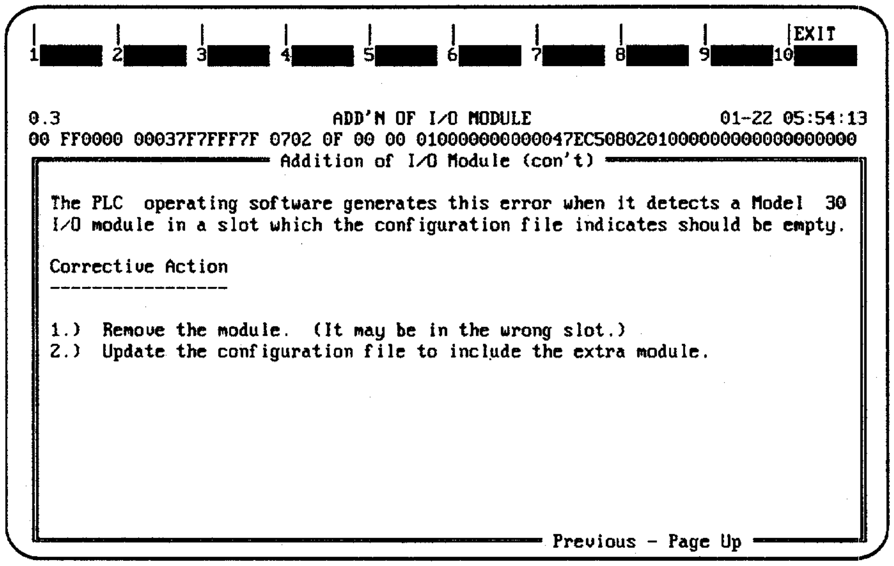


```
|        |        |        |        |        |        |        |        |        |EXIT
1        2        3        4        5        6        7        8        9        10

0.3                            ADD'N OF I/O MODULE                        01-22 05:54:13
00 FF0000 00037F7FFF7F 0702 0F 00 00 0100000000000047EC50802010000000000000000
                         Addition of I/O Module

  The PLC operating  software  generates this error when an I/O module, which
  had been faulted, returns to operation.

  Corrective Action
  -----------------

  1.)  No  action  necessary  if  the  module was removed or replaced, or the
       remote rack was power cycled.
  2.)  Update the configuration file or remove the module.

              Next - Page Down
```

Note

A hexadecimal dump of the fault can be displayed by pressing **CTRL-F** from this screen. For more information on fault explanations and correction and **CTRL-F**, please refer to the *Series 90-30/20/Micro Programmable Controllers Reference Manual,* GFK-0467.

Press the **Page Down** key to display additional data pertaining to the fault.



```
0.3                        ADD'N OF I/O MODULE                    01-22 05:54:13
00 FF0000 00037F7FFF7F 0702 0F 00 00 0100000000000047EC5080201000000000000000000
                       Addition of I/O Module (con't)

The PLC  operating software generates this error when it detects a Model  30
I/O module in a slot which the configuration file indicates should be empty.

Corrective Action
-----------------

1.)  Remove the module.  (It may be in the wrong slot.)
2.)  Update the configuration file to include the extra module.

                                                       Previous - Page Up
```

For more information on fault explanations and correction, refer to chapter 3, "Fault Explanation and Correction," in the *Series 90-30/20/Micro Programmable Controllers Reference Manual,* GFK-0467.

All softkeys are disabled on this screen, except for **EXIT** (**Shift-F10**). The **Abort** key (**ALT-A**) and the **Escape** key may also be used to exit from this screen and return to the PLC fault table.

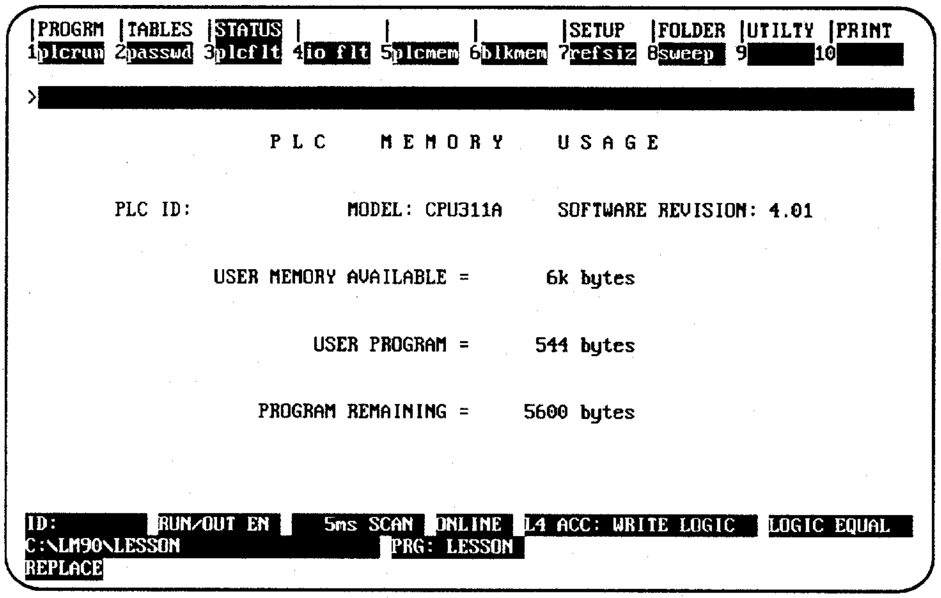
Clearing the Fault Table

You can clear the fault table in **ONLINE** mode by pressing **Clear** (**F9**). (This may be password protected.) Faults can also be cleared from the program logic.

Clearing the fault table removes the faults it contains. Clearing the fault table does not clear fault conditions in the system. If the condition that caused a fault still exists, the fault may be reported again after storing the configuration, cycling power to the PLC, or during a STOP-to-RUN transition.

PLC Memory Used

The PLC Memory Usage screen shows the amount of PLC memory available and the amount used for application program information. To display this screen, press **PLC Memory** (**F5**) from the PLC Control and Status menu or from another PLC functions screen. This is an example of the PLC Memory Usage screen:

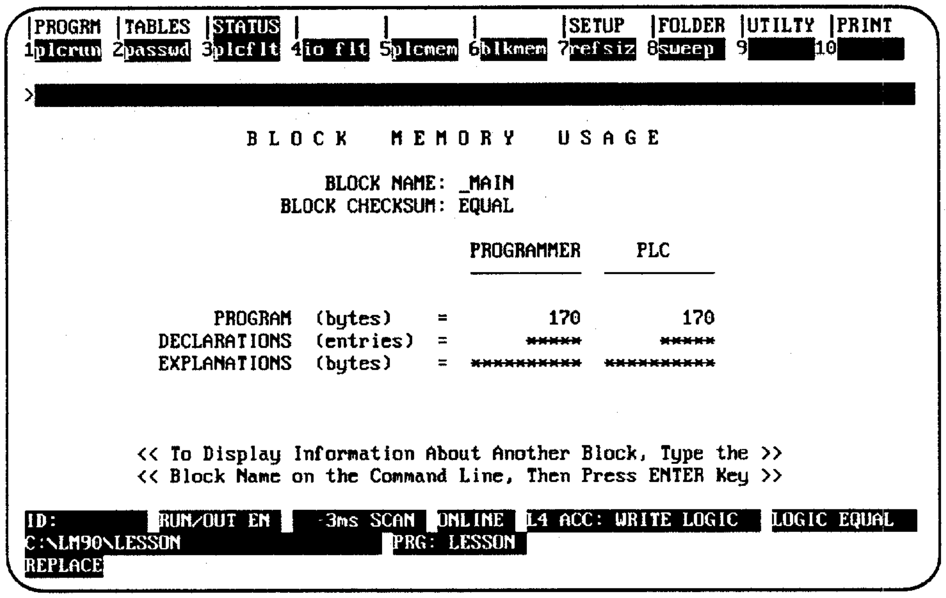


```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1plcrun 2passwd 3plcflt 4io flt 5plcmem 6blkmem 7refsiz 8sweep  9       10
>

                      P L C    M E M O R Y    U S A G E

        PLC ID:               MODEL: CPU311A     SOFTWARE REVISION: 4.01

                    USER MEMORY AVAILABLE =       6k bytes

                             USER PROGRAM =      544 bytes

                        PROGRAM REMAINING =     5600 bytes


ID:           RUN/OUT EN    5ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                       PRG: LESSON
REPLACE
```

Field	Description
PLC ID	In **ONLINE** or **MONITOR** mode, the name of the PLC being communicated with. It will always be the same as that shown in the status area of the screen. The ID name can be set on the Assign PLC ID screen in the configuration software.
Model	The model number of the attached PLC: • **CPU 211:** Series 90-20 PLC Model 211. • **UDR1/2:** Series 90 Micro PLC Model UDR001 or UDR002. • **UAA003:** Series 90 Micro PLC Model UAA003 • **UDD004:** Series 90 Micro PLC Model UDD004 • **UDR005:** Series 90 Micro PLC Model UDR005 • **CPU 311:** Series 90-30 PLC Model 311 with a 5 or 10-slot rack. • **CPU 331:** Series 90-30 PLC Model 331 with a 10-slot rack. • **CPU 340:** Series 90-30 PLC Model 340 • **CPU 341:** Series 90-30 PLC Model 341 • **CPU 351:** Series 90-30 PLC Model 351
Software Revision	The revision of PLC software.
User Memory Available	The amount of memory in the PLC that is available for application program information.
User Program	The amount of program memory occupied by the logic program.
ProgramRemaining	The amount of program memory remaining.

Block Memory Usage

The Block Memory Usage screen shows the amount of memory used for different parts of the application program. For example:



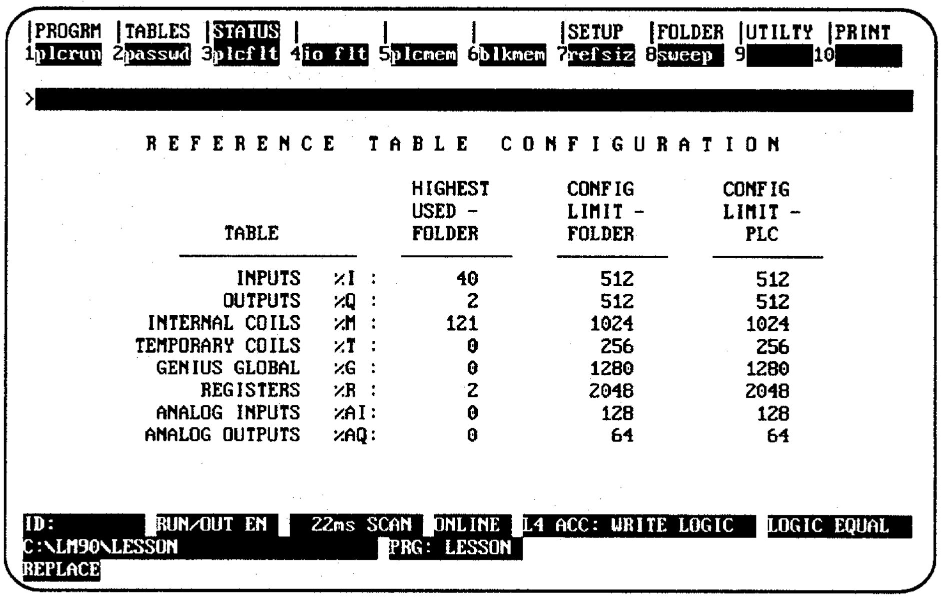
```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1plcrun 2passwd 3plcflt 4io flt 5plcmem 6blkmem 7refsiz 8sweep  9       10
>

                   B L O C K    M E M O R Y    U S A G E

                           BLOCK NAME: _MAIN
                       BLOCK CHECKSUM: EQUAL

                                       PROGRAMMER     PLC
                                       __________   __________

                  PROGRAM  (bytes)    =       170         170
             DECLARATIONS  (entries)  =     *****       *****
             EXPLANATIONS  (bytes)    =  **********  **********


          << To Display Information About Another Block, Type the >>
          << Block Name on the Command Line, Then Press ENTER Key >>

ID:          RUN/OUT EN    3ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                        PRG: LESSON
REPLACE
```

To display the Block Memory Usage screen, press **Block Memory** (**F6**) from the PLC Control and Status menu or from another PLC functions screen. The first block for which information appears is the _MAIN block. To display information for another subroutine block, enter its name on the command line and press the **Enter** key.

Field	Description
Block Name	The name of the block for which values are displayed. Values can also be displayed for locked blocks.
Block Checksum	The result of a comparison of the code checksums for the block from the programmer and from the PLC. This is either EQUAL or NOT EQUAL.
Program	The block's logic memory size.
Declarations	The block's symbol table size.
Explanations	The block's explanation text size. This field will always display asterisks since no explanations are currently stored in the PLC.

Configured Reference Sizes

To see how much memory has been used for program references, press **Reference Size** (**F7**) from the PLC Control and Status menu or from another PLC functions screen.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1plcrun 2passwd 3plcflt 4io flt 5plcmem 6blkmem 7refsiz 8sweep  9       10
>

          R E F E R E N C E   T A B L E   C O N F I G U R A T I O N

                                  HIGHEST      CONFIG       CONFIG
                                  USED -       LIMIT -      LIMIT -
                TABLE             FOLDER       FOLDER       PLC

                 INPUTS   %I :       40           512          512
                OUTPUTS   %Q :        2           512          512
         INTERNAL COILS   %M :      121          1024         1024
        TEMPORARY COILS   %T :        0           256          256
          GENIUS GLOBAL   %G :        0          1280         1280
              REGISTERS   %R :        2          2048         2048
          ANALOG INPUTS   %AI:        0           128          128
         ANALOG OUTPUTS   %AQ:        0            64           64


ID:            RUN/OUT EN   22ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                         PRG: LESSON
REPLACE
```

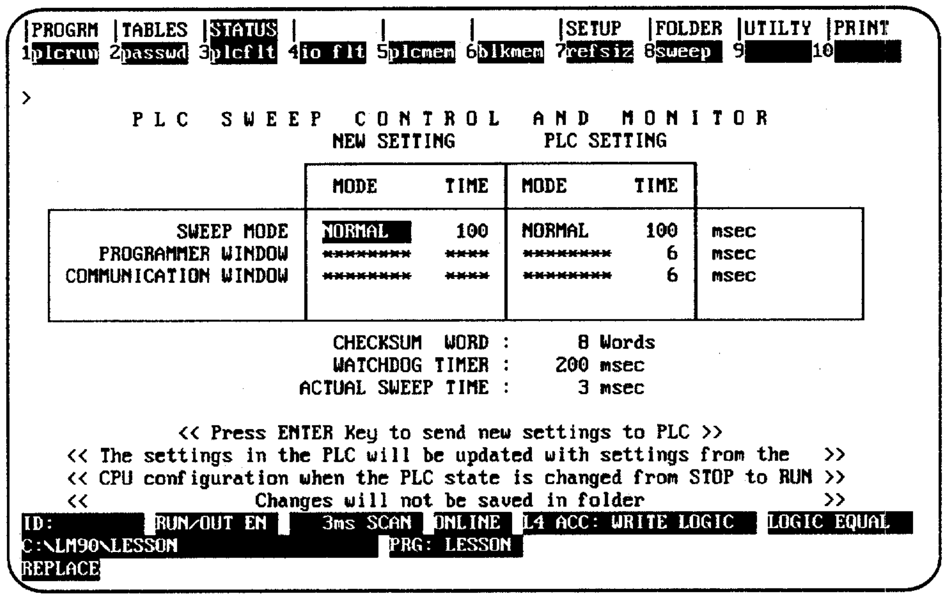
This function is available in all programmer operating modes and also to locked blocks. No values are displayed for the PLC if the programmer is in **OFFLINE** mode. No values are displayed for the configuration file if there is no configuration file in the current folder. Values that do not match are highlighted.

Field	Description
Highest Used – Folder	Lists the highest value used.
Config Limit – Folder	Lists a configuration value for each reference type in the folder.
Config Limit – PLC	Lists a configuration value for each reference type in the PLC.

PLC CPU Sweep Control

You can set the Sweep while the PLC is in RUN mode, and it only affects the PLC during that RUN; i.e., each time the PLC goes from STOP to RUN mode, the default sweep mode takes effect. No matter what the default sweep setting is, you can change it for the current RUN and have the resulting effects immediately applied. You can use the Active Constant Sweep Mode Parameter to toggle the sweep mode of the PLC without having to change the configured sweep settings, or you can change the amount of time for the constant sweep on this same screen. (This may be very useful when you need to fine tune the sweep time while the PLC is running a program.)

To display or change PLC timers, press **sweep** (**F8**) from the PLC Control and Status menu or from another PLC functions screen.



```
|PROGRM |TABLES |STATUS |        |        |        |SETUP  |FOLDER |UTILTY |PRINT
1plcrun 2passwd 3plcflt 4io flt 5plcmem 6blkmem 7refsiz 8sweep  9       10

>
        P L C   S W E E P   C O N T R O L   A N D   M O N I T O R
                           NEW SETTING          PLC SETTING
                      MODE        TIME   MODE        TIME
            SWEEP MODE  NORMAL       100   NORMAL       100   msec
     PROGRAMMER WINDOW  ********    ****   ********       6   msec
  COMMUNICATION WINDOW  ********    ****   ********       6   msec

                       CHECKSUM  WORD :       8 Words
                       WATCHDOG TIMER :     200 msec
                    ACTUAL SWEEP TIME :       3 msec

              << Press ENTER Key to send new settings to PLC >>
     << The settings in the PLC will be updated with settings from the   >>
     << CPU configuration when the PLC state is changed from STOP to RUN >>
     <<             Changes will not be saved in folder                  >>
ID:          RUN/OUT EN     3ms SCAN  ONLINE  L4 ACC: WRITE LOGIC    LOGIC EQUAL
C:\LM90\LESSON                        PRG: LESSON
REPLACE
```

In **ONLINE** or **MONITOR** mode, the software shows the current CPU sweep time information.

To change the Active Constant Sweep Mode Parameter, press the **Tab** key to change the first selection ("Mode") from "NORMAL" to "CNST SWP" (if necessary—if you are just changing the time of the Constant Sweep, it will already say "CNST SWP"). Press the **Right Cursor** (arrow) key to advance the cursor to the "Time" selection. Then enter the desired number of milliseconds (5 to 200 with a default of 100 for most CPUs—5 to 500 for the 351 CPU).

Note

Remember that the number of seconds entered here cannot exceed the value of the Watchdog Timer. Also remember that the change you make through the Active Constant Sweep Mode Parameter will have immediate effect but will be replaced by the default sweep mode each time the PLC goes from STOP to RUN mode.

Press the **Escape** key when done.

Note

Remember, settings from this screen are only stored in the PLC, not in the folder. These settings may be configured and stored in the folder using the configuration function. For more information, refer to chapter 10, section 3, "Configuring the CPU Module."

Active Constant Sweep Mode Setting

No matter what the default sweep setting is, you can change it when the PLC is in **RUN** mode and have the resulting effects immediately applied. You can use the Active Constant Sweep Mode Parameter to toggle the sweep mode of the PLC without having to change the configured sweep settings, or you can change the amount of time for the constant sweep on this same screen. (This may be very useful when you need to fine tune the sweep time while the PLC is running a program.)

To change the constant sweep time (unless it is password-protected) or the mode, Logicmaster must be in **ONLINE** mode. To set the Active Constant Sweep Mode Parameter, after entering the PLC Sweep Control screen, press the **Tab** key to change the first selection "Sweep Mode" from "NORMAL" to "CNST SWP" (if necessary—if you are just changing the time of the Constant Sweep, it will already say "CNST SWP"). Press the Right Cursor (arrow) key to advance the cursor to the "Time" selection. Enter the desired number of milliseconds (5 to 200 with a default of 100).

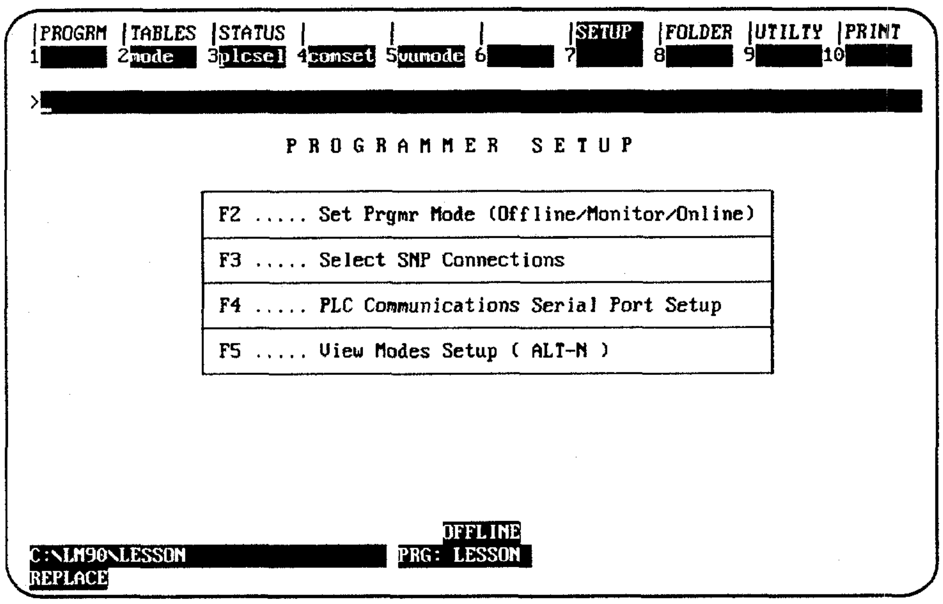
Note

Remember that the number of seconds entered here cannot exceed the value of the Watchdog Timer. Also remember that the change you make through the Active Constant Sweep Mode Parameter will have immediate effect but will be replaced by the default sweep mode each time the PLC goes from STOP to RUN mode.

For an explanation of the PLC sweep, refer to chapter 2, "System Operation," in the *Series 90-30/20/Micro Programmable Controllers Reference Manual,* GFK-0467.

Chapter 6 Programmer Setup

This chapter explains how to set up the programmer for communication with the PLC and how to select the programmer operating mode. To use the programmer setup features, press **SETUP (F7)** from within the programming software.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1       2mode   3plcsel 4comset 5vumode 6       7       8       9       10
>_

                    P R O G R A M M E R   S E T U P

            F2 ..... Set Prgmr Mode (Offline/Monitor/Online)
            F3 ..... Select SNP Connections
            F4 ..... PLC Communications Serial Port Setup
            F5 ..... View Modes Setup ( ALT-M )

                                          OFFLINE
C:\LM90\LESSON                            PRG: LESSON
REPLACE
```

Note

On computers with a keyswitch, such as the Workmaster or CIMSTAR I industrial computer, the **Mode Selection (F2)** key does not appear on the Programmer Setup screen, unless the keyswitch was disabled during programmer setup.

Section 1: Programmer Operating Mode

During configuration and programming, the computer is always in one of three operating modes:

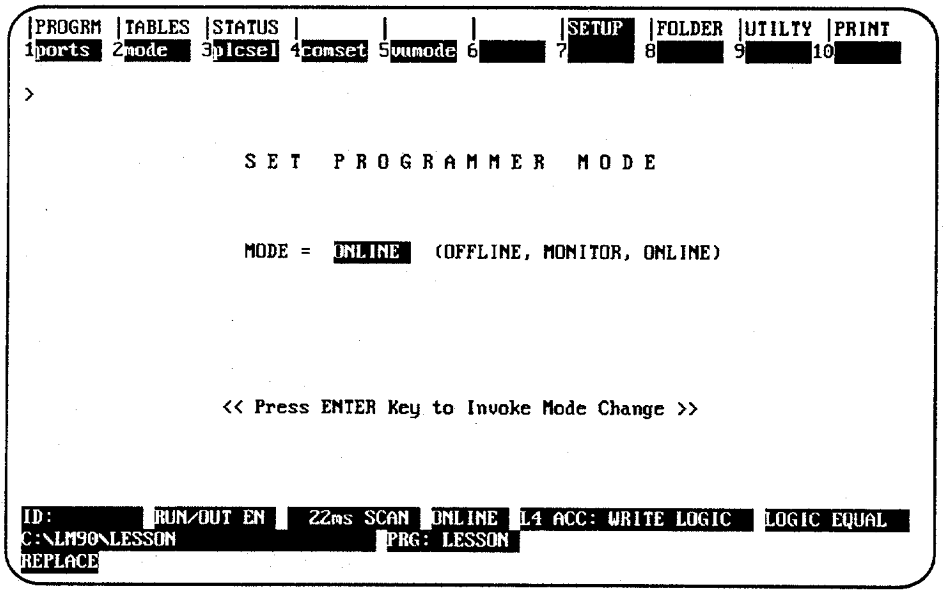
Mode	Description
OFFLINE	**OFFLINE** mode is used for program development. The programmer does not communicate with the PLC in **OFFLINE** mode; power flow display and reference values are not updated.
ONLINE	**ONLINE** mode provides full CPU communications, allowing data to be both read and written.
MONITOR	**MONITOR** mode allows programs to be examined and real-time status to be displayed, but no changes of logic, reference values or I/O overrides are allowed. For Workmaster and CIMSTAR I computers, **MONITOR** mode is the only mode which allows the key to be removed from the keyswitch.

Many functions require the computer to be in either **MONITOR** or **ONLINE** mode. To use either of these operating modes, communications must have been established between the computer and the CPU.

Mode Selection

Both the Workmaster and CIMSTAR I computers have a keyswitch which must be used to select the programmer operating mode. The presence of this keyswitch must be indicated in the programmer setup file, as described in appendix C.

For computers without a keyswitch (e.g., the Workmaster II computer), the programmer mode can be selected by pressing **Mode** (**F2**) from the Programmer Setup menu or the Serial Port Setup screen. The programmer mode can also be changed by pressing **ALT-M** from another programming function.

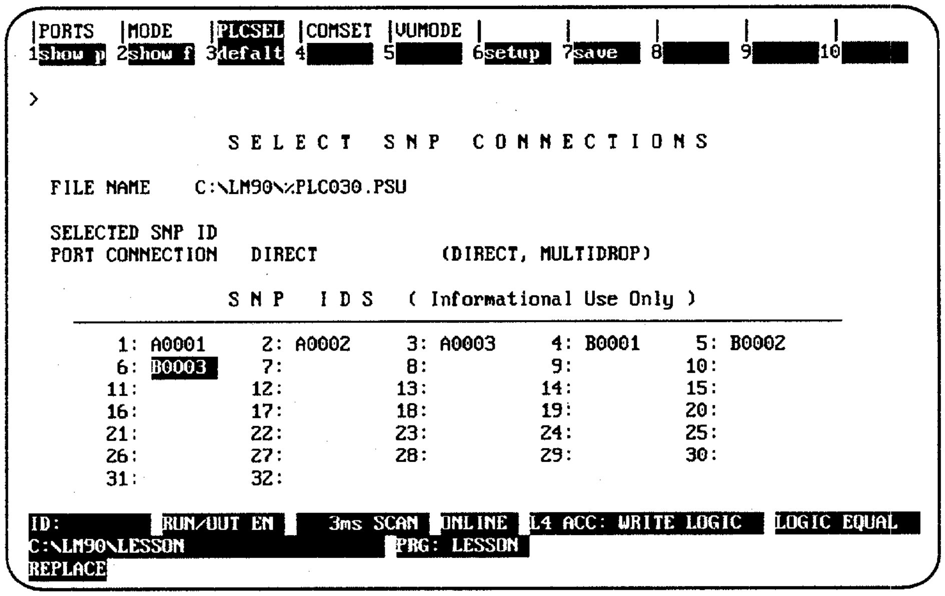


```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1ports  2mode   3plcsel 4comset 5vumode 6       7       8       9       10

>

                    S E T   P R O G R A M M E R   M O D E


                    MODE =  ONLINE   (OFFLINE, MONITOR, ONLINE)



                 << Press ENTER Key to Invoke Mode Change >>


ID:          RUN/OUT EN    22ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                         PRG: LESSON
REPLACE
```

To change the programmer operating mode, use the Tab key to toggle choices at the cursor position or type in a new value. Press the **Enter** key.

Section 2: Selecting SNP Connections

The "Select SNP Connections" screen provides a log of SNP IDs defined by the user. Logicmaster 90-30/20/Micro software does not create or use the information in these fields, but it may be helpful to refer to this listing when selecting a specific SNP ID.



```
|PORTS   |MODE    |PLCSEL  |COMSET  |UUMODE  |        |        |        |        |
1show p  2show f  3defalt  4        5        6setup   7save    8        9        10

>

                 S E L E C T   S N P   C O N N E C T I O N S

  FILE NAME     C:\LM90\%PLC030.PSU

  SELECTED SNP ID
  PORT CONNECTION   DIRECT          (DIRECT, MULTIDROP)

                 S N P   I D S   ( Informational Use Only )

        1: A0001     2: A0002     3: A0003     4: B0001     5: B0002
        6: B0003     7:           8:           9:          10:
       11:          12:          13:          14:          15:
       16:          17:          18:          19:          20:
       21:          22:          23:          24:          25:
       26:          27:          28:          29:          30:
       31:          32:

ID:            RUN/OUT EN     3ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                          PRG: LESSON
REPLACE
```

Field	Description
File Name	The name of a disk file containing the PLC selection information. If no file name is entered, the default name %PLC030.PSU is used. This field allows you to save the setup parameters to a file other than the default file name. You can then recall the setup from this saved file and perform the setup within that screen.
Selected SNP ID	The name used to identify the PLC to be communicated with. A null string may be specified as the SNP ID by leaving the field blank. This allows communication to any PLC using the currently active serial port setup. The null string SNP ID should be used only for direct (point-to-point) PLC connections, since all PLCs will respond in a multidrop configuration.
Port Connection	The serial link configuration. The serial port can be set up for direct (point-to-point) communication with one PLC, or multidrop configuration with several PLCs connected to the WSI Board serial port. Direct connection should not be used with a multidrop configuration. In a multidrop configuration, all PLCs must be set up with the same parity and number of stop bits. They can, however, have a different baud rate.
SNP ID Numbers	These fields allow you to enter the SNP IDs which have already been defined.

Section 3: PLC Communications Serial Port Setup

In the standard serial communications version of the software package, the PLC Communications Serial Port Setup screen takes the place of the Programmer WSI Serial Port Setup screen used in the WSI-based version. If you are using the WSI Board for serial communications, skip this information and go to the information on WSI serial port setup which follows.

The Standard Serial COM Port version of Logicmaster 90-30/20/Micro software provides logic programming and configuration for the Series 90-30, 90-20, or Micro PLC, using the COM1, COM2, COM3, or COM4 serial port instead of a Work Station Interface (WSI) board. This version is available as a separate programming package. It also supports offline programming.

Standard Serial COM Port

The contents of the Standard Serial COM Port version of Logicmaster 90-30/20/Micro programming package include:

- One 5.25-inch High Density diskette, two 3.5-inch Double Density diskettes.
- *Logicmaster 90-30/20/Micro Programming Software User's Manual,* GFK-0466.
- *Series 90-30/20/Micro Programmable Controller Reference Manual,* GFK-0467.
- *Important Product Information for the Standard Serial COM Port Version of Logicmaster 90-30/20/Micro Software,* GFK-0683.
- If the programming package is ordered as IC641SWC306 (or IC641SWC307 depending on the type of diskettes used), a Miniconverter Kit, IC690ACC901, is included. The kit consists of an RS-422 (SNP) to RS-232 mini-converter, a 6-foot (2 meter) serial extension cable, and a 9-pin to 25-pin converter plug assembly.

The Standard Serial COM Port version of Logicmaster 90-30/20/Micro software uses a software communications driver to perform the functions of the WSI board. This communications driver, when added to the Logicmaster 90-30/20/Micro software, may not fit in the available memory of most computers *unless* MS-DOS Version 5.0 (or higher) or a commercially available memory manager is used.

The following steps must be performed in order to establish communications with the Series 90-30/20/Micro PLC:

1. Configure the computer memory for the communications driver. Logicmaster 90-30/20/Micro software uses a communications driver that may be loaded into different memory areas. Please refer to "Configuring Memory for the Communications Driver" on page 6-6.
2. If both versions of the software package (WSI or Standard Serial COM Port) are installed on your computer, you must select which version to run from the Start-up Menu's Setup File Editor. (For directions on how to set up your startup menu to handle both versions, refer to page 2-14 and appendix C, "Programmer Environment Setup.")
3. Connect the cable between the computer's serial port (COM1, COM2, COM3, or COM4), the RS-422/RS-485 Converter, and the Series 90-30/20/Micro PLC serial port.

Note

If you have the Miniconverter Kit, IC690ACC901, please refer to GFK-0682 for more information. If you wish to make your own cable, refer to the *Series 90-30/20/Micro Programmable Controller Installation Manual,* GFK-0356, for more information.

4. Set up the computer's serial port to be used for PLC communications. It should match the characteristics of the Series 90-30/20/Micro PLC serial port. Please refer to "Setting up the Computer's Serial Port," on page 6-10.

Requirements

To run the Standard Serial COM Port version of Logicmaster 90-30/20/Micro software, you will need:

Requirement	Description
MS-DOS	MS-DOS Version 5.0 or higher.
Memory Manager	MS-DOS Version 5.0 (or higher) or a memory manager that complies with one of the following specifications: • The LIM/EMS (Lotus Intel Microsoft/Expanded Memory Specification) Version 3.2 or later. • The XMS (eXtended Memory Specification) Version 2.0 or later. • A list of some compliant memory managers is included at the end of this section.
Computer	A Workmaster II, Zenith Mastersport SL Notebook, or other computer with a 80386 or higher processor and at least one available COM port.
Hard Disk	At least 4 Megabytes of hard disk space.
RAM	At least 564 KB (577,536 bytes) of available DOS conventional memory; or 520 KB of available DOS conventional memory *and* 42 KB of High Memory Area, Upper Memory Block, or Expanded Memory, for the COM port driver.
Miniconverter Kit	A Miniconverter Kit, IC690ACC901, may be ordered. This kit contains the necessary cable and converter to connect from your computer to the Series 90 PLC.

Port Requirements

The Standard Serial COM Port version requires at least one Standard Serial COM Port in the host computer in order to provide communications with the PLC. This version will support either the COM1, COM2, COM3, or COM4 port in common use by IBM-compatible machines. These ports must use the INS8250 UART chip, or functional equivalent. The COM port I/O addresses and interrupt requests are shown in the following table:

Port	Port Address	IRQ
COM1	3F8	IRQ4
COM2	2F8	IRQ3
COM3	3E8	IRQ4
COM4	2E8	IRQ3

While the Standard Serial COM Port version of Logicmaster 90-30/20/Micro software is active, the serial port assigned to PLC communications will be used exclusively for PLC communications. If a serial printer is being used, a second serial port is required; or you may direct print output to disk files, and then use MS-DOS to print the disk files while outside the Logicmaster environment.

The Standard Serial COM Port version of Logicmaster 90-30/20/Micro software monitors Clear-to-Send (CTS) to determine a cable disconnect. Therefore, Logicmaster 90-30/20/Micro software must use a connection with the Series 90-30/20/Micro PLC that will keep its Request-to-Send (RTS) active in **ONLINE** or **MONITOR** mode. The RTS at the PLC must be connected to CTS at the Logicmaster computer serial port. Half-duplex modems cannot be used with Logicmaster 90-30/20/Micro software since they rely on RTS/CTS flow control, which is not supported by the software.

The Standard Serial COM Port version of Logicmaster 90-30/20/Micro software supports multi-drop connections, using RS-422 4-wire hookups or modems. RS-422 multi-drop will work as long as the Series 90 PLC slaves transmit only in response to requests from an attached Logicmaster host.

Configuring Memory for the Communications Driver

The Standard Serial COM Port version of Logicmaster 90-30/20/Micro software requires a communications driver that may be loaded in different memory areas. The communications driver requires 42 KB of memory. A memory manager may be required, depending on the computer and the amount of installed memory. That is to say, you can load the communications driver (requiring 42 KB of memory) into conventional memory if you have enough available conventional memory to do so; if not, then you will need to ensure that it can be loaded into one of the other areas of memory described on the next page and discussed in detail on page 6-13 and following.

MS-DOS Memory Areas Defined

One of the following memory areas is used to load the communications driver:

- Extended Memory, consisting of the High Memory Area and extended memory blocks.
- MS-DOS System Memory, consisting of Conventional Memory and Upper Memory.
- Expanded Memory.

Type	Description
High Memory Area	65,520 bytes of memory space just above the 1 Megabyte boundary (addresses 0FFFFF to 10FFFF). This area can only be accessed by computers with 80286/ 80386/80486 microprocessors. PCs with 8086/8088/80188/80186 microprocessors cannot address this memory space. Use of the High Memory Area (HMA) requires either HIMEM.SYS or QEMM-386™ memory manager. The memory manager must comply to the Extended Memory Specification XMS 2.0. For more information on using the High Memory Area on an 80386 (Workmaster II) or higher computer, refer to page 6-14.
Upper Memory Block	Memory space between 640 KB and 1 Megabyte. This memory space is usually used for BIOS or video RAM. On computers with 80386/80486 microprocessors and extended memory installed, extended memory can be mapped into this area with the aid of a memory manager or MS-DOS Version 5.0 or higher. If you want the communications driver loaded into this area, you can configure the memory manager to make this area available. For more information on using the Upper Memory Block on an 80386 (Workmaster II) or higher comput-er., refer to page 6-15.
Conventional Memory	The first 640 KB of memory addresses. It is sometimes referred to as low memory (RAM) or base memory (RAM). For more information on using conventional memory on an 80386 (Workmaster II) or higher computer, refer to page 6-16. For more information on using video RAM on an 80386 (Workmaster II) or higher computer, refer to page 6-17.
Expanded Memory	The user-installed memory option that can be accessed with an expanded memory manager through a page frame area between the MS-DOS 640 KB and 1024 KB addresses (e.g., GEXMEM). The communications driver may be loaded to the page frame area. This memory area is suitable for computers with Intel 80xx family processors. Some laptop personal computers with 8086/8088 microprocessor and 1 Megabyte of memory come with an expanded memory manager driver (for example, Toshiba 1200). The memory managers must comply to the Lotus/Intel/Microsoft LIM Expanded Memory Specification 3.2 or later. For more information on using Expanded Memory on an 80386 (Workmaster II) or higher computer, refer to page 6-16.

™ QEMM-386 is a trademark of Quarterdeck Office Systems.

Communications Driver Load Order

To specify the area for the communications driver, you must enter Setup from the main menu. Refer to appendix C, "Programmer Environment Setup," for instructions on how to change the memory allocation for this driver.

The recommended setting for this driver memory allocation is "Automatic," i.e., you let Logicmaster decide how best to set up the memory allocation. If the option selected from the PLC Communications Options screen is automatic, communications driver is loaded into the first available memory area found, according to the following sequence:

1. If the High Memory Area is free and the computer supports the High Memory Area, the driver is installed in the High Memory Area.
2. If the High Memory Area is not available, Logicmaster 90-30/20/Micro software will look for the Upper Memory Block area. If there is enough space for the driver, the driver can be installed in the Upper Memory Block space.
3. Then, if neither the High Memory Area nor the Upper Memory Block is available, Logicmaster 90-30/20/Micro software will examine the MS-DOS conventional memory. If the MS-DOS conventional memory space is large enough for Logicmaster 90-30/20/Micro software and the communications driver to coexist, the driver will be installed in the MS-DOS memory space as a Terminate-and-Stay-Resident (TSR) program.
4. Next, Logicmaster 90-30/20/Micro software will examine the Expanded Memory. If the expanded memory manager is installed, the loader will try to allocate an expanded memory page frame area and install the driver in the page frame area.
5. If all these steps fail, then the driver is not installed and a message will appear on your screen stating, "PLC communications driver was not installed." In addition, the following prompt will be displayed: "Do you wish to continue? (Y/N)" If you answer: **Y**, you may program offline without PLC communication capability.
6. If you still wish to communicate with the PLC, you must ensure that one of the memory areas listed in steps 1 through 4 above is available. Instructions for memory management are listed on page 6-13 and following of this manual and are also included in your DOS manual. If you have an additional memory manager, such as QEMM, you may wish to refer to that manual as well.

Note

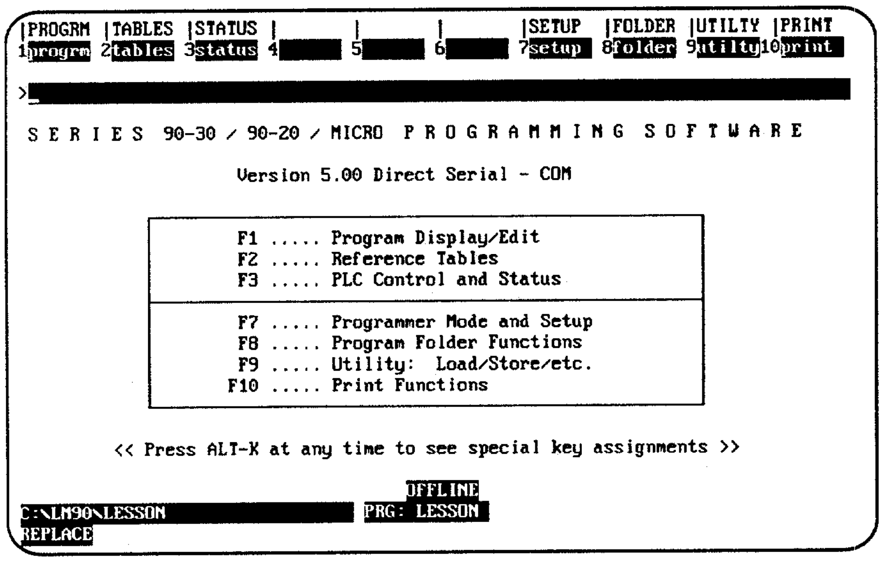
The PLC Communications Serial Port Setup screen in the setup function of Logicmaster 90-30/20/Micro software will display where the communications driver was loaded. Refer to "Setting Up the Computer's Serial Port" on page 6-10 for more information.

Running Logicmaster 90 Software

When entering the Standard Serial COM Port version of the software package, the software will attempt to install the correct communications driver in the available memory space. As mentioned previously, if the driver cannot be installed, the message, "PLC communications driver was not installed," will appear on the Initializing screen.

The software will prompt you to continue. If you enter **N** after the prompt, the Logicmaster 90 main menu is displayed. You may then exit back to MS-DOS to correct the memory problem.

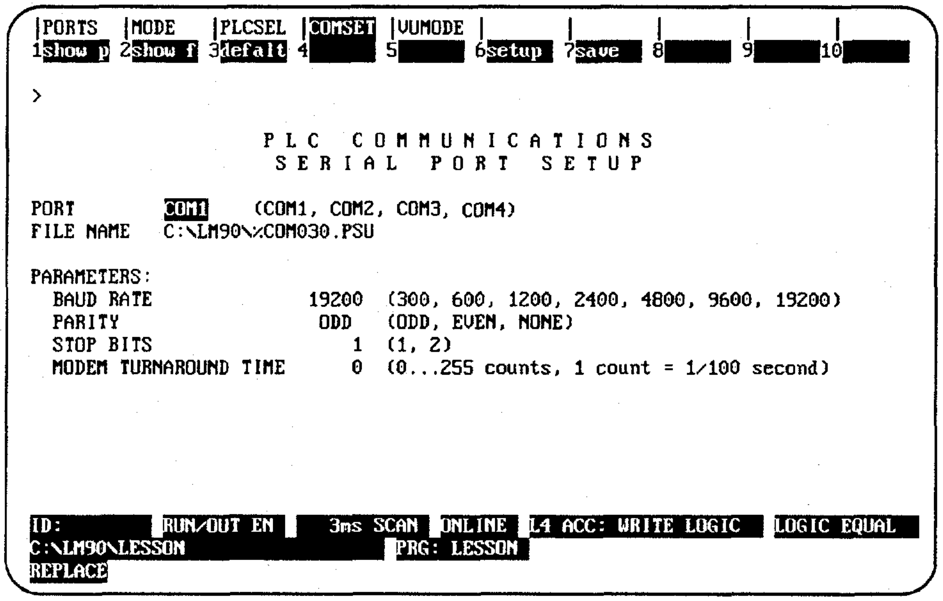
If you enter **Y**, the software will proceed without PLC communication capability. (You may program offline.) The main menu screen is then displayed. For example, the programmer software would display this screen:


|PROGRM |TABLES |STATUS | | | |SETUP |FOLDER |UTILTY |PRINT
1progrm 2tables 3status 4 5 6 7setup 8folder 9utilty 10print
>

```
SERIES 90-30 / 90-20 / MICRO PROGRAMMING SOFTWARE

Version 5.00 Direct Serial - COM

 F1 ..... Program Display/Edit
 F2 ..... Reference Tables
 F3 ..... PLC Control and Status

 F7 ..... Programmer Mode and Setup
 F8 ..... Program Folder Functions
 F9 ..... Utility: Load/Store/etc.
F10 ..... Print Functions

<< Press ALT-K at any time to see special key assignments >>

OFFLINE
C:\LM90\LESSON                     PRG: LESSON
REPLACE
```

Setting Up the Computer's Serial Port

The computer's serial port (COM1, COM2, COM3, or COM4) may be set up from the PLC Communications Serial Port Setup screen in the setup function. Press **COMSET** (**F4**) (*COMSET* is the abbreviation for Communications Setup) from the Programmer Setup menu to display the PLC Communications Serial Port Setup screen shown below:



```
|PORTS  |MODE   |PLCSEL |COMSET |VUMODE |       |       |       |       |
1show p 2show f 3defalt 4       5       6setup  7save   8       9       10

>

                    P L C   C O M M U N I C A T I O N S
                     S E R I A L   P O R T   S E T U P

PORT        COM1    (COM1, COM2, COM3, COM4)
FILE NAME   C:\LM90\%COM030.PSU

PARAMETERS:
  BAUD RATE                 19200  (300, 600, 1200, 2400, 4800, 9600, 19200)
  PARITY                     ODD   (ODD, EVEN, NONE)
  STOP BITS                    1   (1, 2)
  MODEM TURNAROUND TIME        0   (0...255 counts, 1 count = 1/100 second)




ID:          RUN/OUT EN     3ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                        PRG: LESSON
REPLACE
```

Note

The PLC CPU defaults to the parameters displayed on this screen (e.g., 19200 baud, odd parity, etc.). These parameters must match the CPU slot configuration that was last stored to the PLC before communications can be established (see chapter 10, section 3, "Configuring the CPU Module").

The table on the next page (i.e., the facing page) provides detailed information about the fields in the screen shown above.

Field	Description
Port Selection	Select the serial COM port (COM1, COM2, COM3, or COM4) to be used for communicating with the PLC.
Setup File Name	The default file name for saving the setup parameters is changed from %WSI030.PSU to %COM030.PSU, to avoid any confusion with the WSI-based version. The %COM030.PSU will contain additional information, such as the COM port selection, which is not present in the %WSI030.PSU file.
Parameters	Press **Default (F3)** to set the parameters on this screen to their default values. The PLC CPU defaults to these parameters, as displayed on the screen shown above (e.g., 19200 baud, odd parity, 1 stop bit, etc.). These parameters must match the CPU slot configuration that was last stored to the PLC before communications can be established (see chapter 11, section 3, "Configuring the CPU Module"). Baud Rate: The transmission rate, in bits per second, of data through the port. Parity: Specify whether the parity bit is **odd** or **even**; or, if no no parity bit is needed, select **none**. To toggle the selections, place the cursor on the field and press **Tab** to toggle forward (**Shift-Tab** to toggle backwards.) Stop Bits: All serial communication uses at least one stop bit. Slower devices may use two stop bits. Modem Turnaround Time: The time required for the modem to start data transmission after receiving the transmit request. If no modem is used, specify zero. When the PLC is connected through a modem, the value must be one or greater.
PLC Communication Driver Information	Information on the communications driver, including: • Version. • Where the driver was installed: High Memory Area (HMA), Expanded Memory (EMS), Conventional MS-DOS Memory (DOS), or Upper Memory Block (UMB) Area. • MS-DOS segment address at which the driver is loaded. • MS-DOS INT vector that it will use.

Setting Up the Port

To set up the serial port, enter the appropriate values in the fields on the Serial Port Setup screen. Use the **Up** and **Down Arrow** (cursor mover) keys to move from field to field. In each field, you can use the **Tab** key to cycle through the valid selections, **Shift-Tab** to reverse the direction of the selection display. Once all the values have been entered, press **Setup (F6)**.

Displaying Port Settings

Press **Show Port (F1)** to display the current settings for the Serial Port Setup screen.

Pressing **F1** *while completing the screen* will delete all the changes that have been made. The previous settings of the current driver will then be re-displayed on the screen.

Saving the Port Setup

Press **Save (F7)** to save a copy of the current serial port settings to a disk file. The data will be saved to a separate file. If you do not specify a file name, a default name will be assigned.

To create a file of settings which is not automatically invoked, enter a file name other than the default name. If no path is given, the current default directory is used. If no file extension is provided, .PSU is used.

After you press **Save (F7)**, the settings are stored to a file and also set up on the port.

Displaying File Settings

The settings currently stored in a file can be viewed by first entering the file name and then pressing **Show File (F2)**. If no path is specified, the current default directory is used. If no file extension is provided, .PSU is used. If the *File Name* field is blank, the default setup file settings are displayed.

Port Usage Conflicts

The standard serial communications version will reserve one of the COM ports for PLC communications. To avoid any conflicts, you should follow these guidelines:

Conflict	Description
Setup Files	The setup file looked for during system initialization is %COM030.PSU. This file is used to initialize the PLC communications port. If more than one setup file is found for the designated PLC communications port, the PLC communications file %PLC030L.PSU will take precedence.
User Setup	If the Printer Serial Port Setup screen is displayed and you attempt to perform a setup on the designated PLC communications port, a warning is displayed and the setup is not done.
Print Destinations	The Print utility provides several screens for selecting the print output device. Any attempt to select the designated PLC communications port as the output device is denied.

Memory Manager Specifications

Any memory manager used to support the Standard Serial COM Port version of Logicmaster 90-30/20/Micro software must comply with one of the following specifications:

1. LIM/EMS (Lotus Intel Microsoft/Expanded Memory Specification) Version 3.02 or later.
2. XMS (eXtended Memory Specification) Version 2.0 or later.

Compliant Memory Managers

The following are some drivers which comply with the memory manager specifications listed above:

3. QEMM-386 (Quarterdeck Expanded Memory Manager for 386). This driver can be used on computers with 80386/80486 microprocessors. It is available from:

Quarterdeck Office Systems
150 Pico Boulevard
Santa Monica, CA 90405
(213) 392-9701

4. QEMM (Quarterdeck Expanded Memory Manager). This driver can be used on computers with 8086/8088 microprocessors. It can be ordered from same address listed above.
5. HIMEM.SYS (MS-DOS Version 5.0 or higher) from Microsoft Corporation. This Microsoft memory manager complies with XMS specification. It can be used on computers with 80386/80486 microprocessors.
6. EMM386.SYS (MS-DOS Version 5.0 or higher) from Microsoft Corporation. This Microsoft memory manager simulates expanded memory while using extended memory. It can be used on computers with 80386/80486 microprocessors.
7. BlueMAX™ for PS/2 computers and 386MAX™ for 80386 AT-based computers. Both are available from:

Qualitas, Inc.
7101 Wisconsin Avenue, Suite 1386
Bethesda, MD 20814
(301) 907-6700

Restrictions

To avoid conflicts of memory usage between the Standard Serial COM Port version of Logicmaster 90-30/20/Micro software and MS-DOS programs, the following rules must be followed:

1. If the computer is set up to use the High Memory Area memory area, the VDISK program must not be used. Any program using VDISK must remove VDISK before running the Logicmaster 90-30/20/Micro software.
2. Another application cannot use Expanded Memory if the communications driver is loaded into Expanded Memory.

™ BlueMAX and 386MAX are trademarks of Qualitas, Inc.

Using High Memory Area on an 80386 (Workmaster II) Or Higher Computer

Physical Extended Memory for High Memory Area Operation

If not already present, you must add and configure physical memory on your computer to be accessed in memory addresses over 1 Megabyte (FFFFF).

386 Memory Manager

You must install a 386 high memory manager such as Microsoft HIMEM.SYS (MS-DOS Version 5.0 or higher), Quarterdeck QEMM-386 or 386MAX. The memory manager should be compliant with the Extended Memory Specification (XMS) Version 2.0 or later.

You can install HIMEM to allow use of the High Memory Area.

1. Verify that your CONFIG.SYS file contains a device command for the HIMEM.SYS extended memory manager (or another memory manager that conforms to the XMS specification). For example:

```
device=c:\dos\himem.sys
```

2. If you have MS-DOS 5.0 (or higher) and wish to use the High Memory Area, do not load MS-DOS into the High Memory Area. (The CONFIG.SYS file should not have a DOS=HIGH command.)
3. Restart your computer by pressing **CTRL-ALT-Delete**.

Using Upper Memory Block on an 80386 (Workmaster II) Or Higher Computer

Physical Extended Memory for Upper Memory Block Operation

If not already present, you must add and configure physical memory on your computer to be accessed in memory addresses over 1 Megabyte (FFFFF).

386 Memory Manager

You should install a 386 memory manager such as Microsoft EMM386 (MS-DOS Version 5.0 or higher), Quarterdeck QEMM-386 or 386MAX. The memory manager must be compliant with the Extended Memory Specification (XMS) Version 2.0 or later.

With MS-DOS 5.0 (or higher), you can install HIMEM and EMM386 to allow use of the Upper Memory Block.

1. Verify that your CONFIG.SYS file contains a device command for the HIMEM.SYS extended memory manager (or another memory manager that conforms to the XMS specification). For example:

```
device=c:\dos\himem.sys
```

2. Add a device command for EMM386 to your CONFIG.SYS file. The device command for EMM386 must come after the device command for HIMEM and before any commands for device drivers that use expanded memory. For example:

```
device=c:\dos\emm386.exe 1024 ram
```

Note

With MS-DOS 6.0, more upper memory can be obtained from B000 – B7FF that is reserved for monochrome VGA display memory; however, this address space is unused on most computers. To enable extra UMBs, include this address range using the following command line instead of the one above:

```
device=c:\dos\emm386.exe 1024 ram i=b000-b7ff
```

3. Add a DOS=HIGH command to the CONFIG.SYS file. For example:

```
dos=high,umb
```

4. Restart your computer by pressing **CTRL-ALT-Delete**.

Using Conventional Memory on an 80386 (Workmaster II) Or Higher Computer

Extended Memory and MS-DOS 5.0 or Higher

If you have MS-DOS 5.0 (or higher), you can load MS-DOS into high memory. This will free more conventional memory for Logicmaster 90-30/20/Micro software and the communications driver. You must have at least 564 KB (577,536 bytes) of available DOS conventional memory.

1. Verify that your CONFIG.SYS file contains a device command for the HIMEM.SYS (MS-DOS Version 5.0 or higher) extended memory manager and the command to load MS-DOS into high memory. For example:

```
device=c:\dos\himem.sys
dos=high
```

2. Restart your computer by pressing **CTRL-ALT-Delete**.

Using Expanded Memory on an 80386 (Workmaster II) Or Higher Computer

Extended Memory

You must install a 386 memory manager such as Microsoft EMM386 (MS-DOS Version 5.0 or higher), Quarterdeck QEMM-386 or 386MAX. These memory managers use extended memory to simulate expanded memory. The memory manager should be compliant with the Extended Memory Specification (XMS) Version 2.0 or later.

386 Memory Manager

To install EMM386 as an expanded-memory emulator:

1. Verify that your CONFIG.SYS file contains a device command for the HIMEM.SYS extended memory manager (or another memory manager that conforms to the XMS specification). For example:

```
device=c:\dos\himem.sys
```

2. Add a device command for EMM386 to your CONFIG.SYS file. The device command for EMM386 must come after the device command for HIMEM and before any commands for device drivers that use expanded memory. For example:

```
device=c:\dos\emm386.exe 1024 ram
```

3. Disable or remove any other device commands for expanded memory managers.
4. Restart your computer by pressing **CTRL-ALT-Delete**.

Using Video RAM on an 80386 (Workmaster II) or Higher Computer

As described above, the QEMM VIDRAM program from Quarterdeck Office Systems can use the EGA/VGA graphic display memory to gain additional MS-DOS conventional memory for use by the communications driver. The graphic display memory (if your EGA/VGA has 64 KB to 96 KB graphic memory) can be turned into 64 KB to 96 KB contiguous memory following the 640 KB conventional memory.

1. Verify that your CONFIG.SYS file contains a device command for the QEMM memory manager. For example:

   ```
   device=c:\qemm\qemm386.sys
   ```

2. Verify that your AUTOEXEC.BAT file contains a command for the VIDRAM program. For example:

   ```
   c:\qemm\vidram on
   ```

Tested Configurations

The configurations listed below have been tested using the Standard Serial COM Port version of Logicmaster 90-30/20/Micro software. GE Fanuc offers this list as a guide to selecting compatible hardware for this version of software. These products have demonstrated compatibility with GE Fanuc via in-house testing and/or customer reports; however, hardware manufacturers' modifications to their products may affect compatibility. This list is not exhaustive and is offered as a guide only. No endorsement of any particular product is intended. If you have any questions, please contact the GE Fanuc Hotline, 1-800-828-5747.

Computer	MS-DOS / Memory Manager	CONFIG.SYS File *

Zenith Data Systems
Z-Note 433 Lnc+
(Driver in MS-DOS)
	MS-DOS 6.0
	Microsoft HIMEM
		device=c:\dos\himem.sys
		device=c:\dos\emm386.exe 1024 ram
		dos=high

Zenith Data Systems
Z-Note 433 Lnc+
(Driver in UMB)
	MS-DOS 6.0
	Microsoft HIMEM
		device=c:\dos\himem.sys
		device=c:\dos\emm386.exe 1024 ram
		dos=high,umb

Zenith Data Systems
Z-Note 433 Lnc+
(Driver in EMS)
	MS-DOS 6.0
	Microsoft HIMEM
		device=c:\dos\himem.sys 1024 ram
		device=c:\dos\emm386.exe
		dos=high

Gateway 2000
4DX-33
(Driver in MS-DOS)
	MS-DOS 6.0
	Microsoft HIMEM
		device=c:\dos\himem.sys 1024 ram
		device=c:\dos\emm386.exe
		dos=high

Gateway 2000
4DX-33
(Driver in UMB)
	MS-DOS 6.0
	Microsoft HIMEM
		device=c:\dos\himem.sys
		device=c:\dos\emm386.exe 1024 ram i=b000-b7ff
		dos=high,umb

Gateway 2000
4DX-33
(Driver in EMS)
	MS-DOS 6.0
	Microsoft HIMEM
		device=c:\dos\himem.sys
		device=c:\dos\emm386.exe 1024 ram
		dos=high

Workmaster II
(Driver in MS-DOS)
	MS-DOS 5.0
	Microsoft HIMEM
		device=c:\dos\himem.sys
		device=c:\dos\emm386.exe 1024 ram
		dos=high

Workmaster II
(Driver in UMB)
	MS-DOS 5.0
	Microsoft HIMEM
		device=c:\dos\himem.sys
		device=c:\dos\emm386.exe 1024 ram
		dos=high,umb

Workmaster II
(Driver in EMS)
	MS-DOS 5.0
	Microsoft HIMEM
		device=c:\dos\himem.sys
		device=c:\dos\emm386.exe 1024 ram
		dos=high

ZEOS
486
(Driver in MS-DOS)
	MS-DOS 6.0
	QEMM 6.0
		device=c:\dos\qemm386.sys
		dos=high

ZEOS
486
(Driver in UMB)
	MS-DOS 6.0
	QEMM 6.0
		device=c:\dos\qemm386.sys ram=D000-Dfff
		dos=high

* Each CONFIG.SYS file contains the lines: files=20 and buffers=48.

Note

If your computer has a WSI card, the switch **`x=ce00-cfff`** should be added to the **`device=c:\dos\emm386.exe`** lines above to reserve this area of memory for its use. For optimum performance, SMARTDRV should be used.

Setting up a Port

To set up the serial port, enter the appropriate values in the fields on the Serial Port Setup screen. Once all the values have been entered, press **Setup** (**F6**).

Displaying Port Settings

The current settings for the Serial Port Setup screen can be displayed by pressing **Show Port** (**F1**).

Pressing **F1** while completing the screen will delete all the changes that have been made. The previous settings of the current driver will then be redisplayed on the screen.

Note

The Standard Serial COM Port version of Logicmaster 90-30/20/Micro software requires either the COM1, COM2, COM3, or COM4 as the communications port for communicating with the Series 90-30/20/Micro PLC. ***Do not use the same COM port for printing*** that you are using for communicating with the Series 90-30/20/Micro PLC. For additional information, see the "Serial Printer Setup" section at the end of this chapter.

Saving the Port Setup

A copy of the current serial port settings may be saved to a disk file by pressing **Save** (**F7**). The data is saved to a separate file. If no file name is specified, a default name is assigned.

To create a file of settings which is not automatically invoked, a file name other than the default name can be specified. If no path is given, the current default directory is used. If no file extension is provided, .PSU is used.

When **Save** (**F7**) is pressed, the settings are stored to a file and also set up on the port.

Displaying File Settings

The settings currently stored in a file can be viewed by first entering the file name and then pressing **Show File** (**F2**). If no path is specified, the current default directory is used. If no file extension is provided, .PSU is used. If the *File Name* field is blank, the default setup file settings are displayed.

Port Usage Conflicts

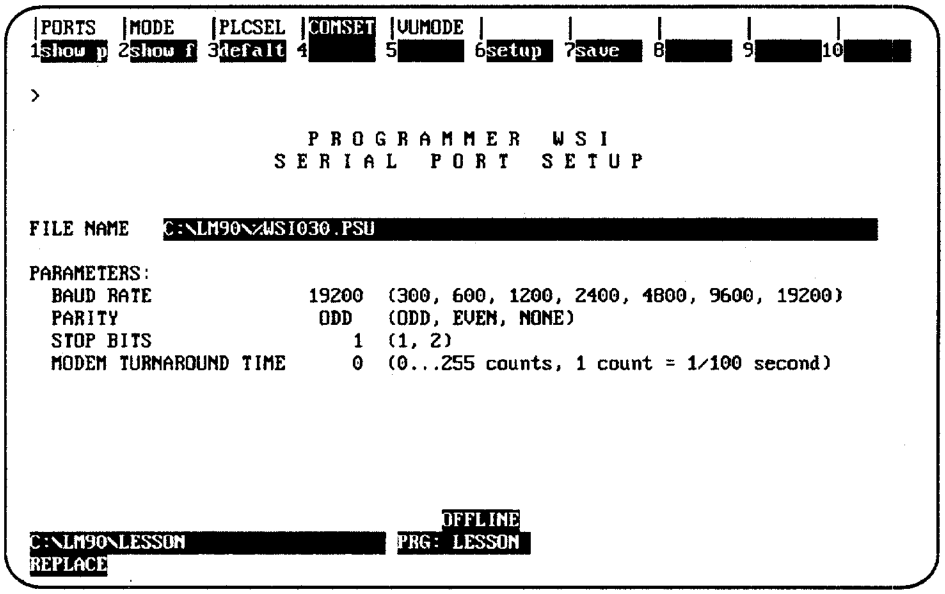
Since the standard serial communications version will reserve one of the COM ports for PLC communications, the following guidelines should be followed:

Type of Conflict	Description
Setup Files	During system initialization, for the standard serial communications version, the setup file looked for is %COM030.PSU. This file is used to initialize the PLC communications port. For the WSI version, the file used is %WSI030.PSU.
User Setup	If the Printer Serial Port Setup screen is displayed and you attempt to perform a setup on the designated PLC communications port, a warning is displayed and the setup is not done.
Print Destinations	The Print utility provides several screens for selecting the print output device. Any attempt to select the designated PLC communications port as the output device is denied.

Section 4: WSI Serial Port Setup

The serial port on the Work Station Interface (WSI) Board in the programmer provides serial communication between the programmer and the attached PLC. The WSI serial port setup functions are used to configure the WSI serial port, and to save or recall those configurations from disk files. If you do not have a WSI Board for serial communications, turn to the information on PLC communications serial port setup which preceded this information.

To display the WSI Serial Port Setup screen, press **COMSET** (**F4**) from the Programmer Setup menu.



```
|PORTS  |MODE   |PLCSEL |COMSET |VUMODE |       |       |       |       |
1show p 2show f 3defalt 4       5       6setup  7save   8       9       10

>

                   P R O G R A M M E R   W S I
                  S E R I A L   P O R T   S E T U P

FILE NAME    C:\LM90\WSI030.PSU

PARAMETERS:
  BAUD RATE               19200  (300, 600, 1200, 2400, 4800, 9600, 19200)
  PARITY                   ODD   (ODD, EVEN, NONE)
  STOP BITS                  1   (1, 2)
  MODEM TURNAROUND TIME      0   (0...255 counts, 1 count = 1/100 second)



                                         OFFLINE
C:\LM90\LESSON                        PRG: LESSON
REPLACE
```

Note

The PLC CPU defaults to the parameters displayed on this screen (e.g., 19200 baud, odd parity, one stop bit, etc.). These parameters must match the CPU slot configuration that was last stored to the PLC before communications can be established (see chapter 10, section 3, "Configuring the CPU Module").

The WSI serial port settings may be viewed, changed, saved to a disk file, or recalled from a disk file in all modes of operation by pressing the appropriate function key.

Field	Description
File Name	The name of a disk file containing the WSI port setup parameters. If no file name is entered, the default name %WSI030.PSU is used. This field allows you to save the setup parameters to a file other than the default file name. You can then recall the setup from this saved file and perform the setup within that screen.
Baud Rate	The transmission rate, in bits per second, of data through the port.
Parity	Specify whether the parity bit is **odd** or **even**; or, if no no parity bit is needed, select **none**. To toggle the selections, place the cursor on the field and press **Tab** to toggle forward (**Shift-Tab**) to toggle backward.
Stop Bits	All communications use one stop bit. Slower devices may use two stop bits.
Modem Turnaround Time	The time required for the modem to start data transmission after receiving the transmit request. If no modem is used,specify zero. When the PLC is connected through modem, the value <u>**must**</u> be one or greater.

Setting Up a Port

To set up the serial port, enter the appropriate values in the fields on the WSI Serial Port Setup screen. Once all the values have been entered, press **Setup** (**F6**).

Displaying Port Settings

The current settings for the WSI Serial Port Setup screen can be displayed by pressing **Show Port** (**F1**).

Pressing **F1** while completing the screen will delete all the changes that have been made. The previous settings of the current driver will then be redisplayed on the screen.

Saving the Port Setup

A copy of the current serial port settings may be saved to a disk file by pressing **Save** (**F7**). The data is saved to a separate file. If no file name is specified, a default name is assigned.

To create a file of settings which will not be automatically invoked, a file name other than the default name can be specified. If no path is given, the current default directory is used. If no file extension is provided, .PSU is used.

When **Save** (**F7**) is pressed, the settings are stored to a file and also set up on the port.

Displaying File Settings

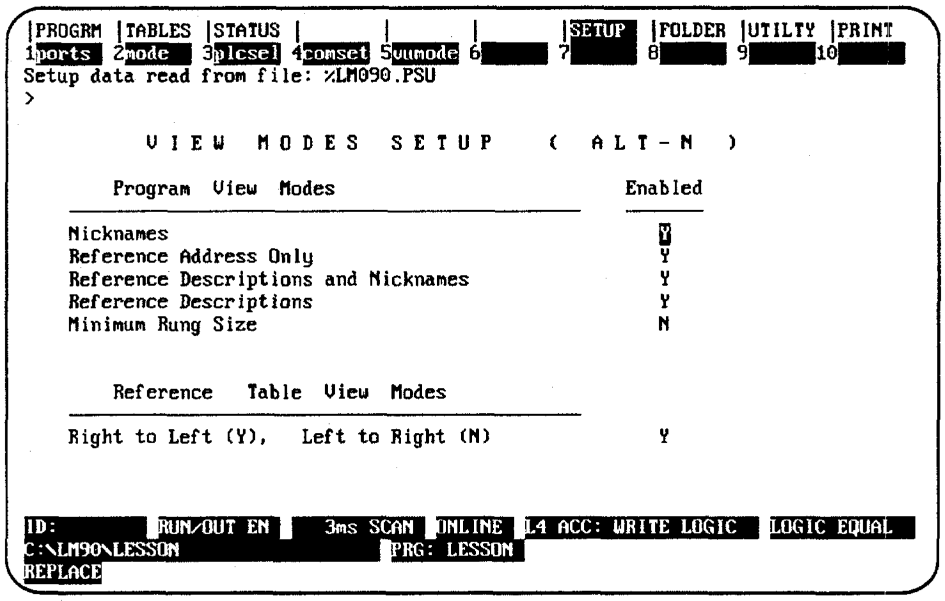
The settings currently stored in a file can be viewed by first entering the file name and then pressing **Show File** (**F2**). If no path is specified, the current default directory is used. If no file extension is provided, .PSU is used. If the *File Name* field is blank, the default setup file settings are displayed.

Section 5: View Modes Setup (ALT-N)

The view mode setup function enables you to specify which modes are displayed when you press **ALT-N**. These view modes range from showing only rung references to showing reference names and reference descriptions in an expanded rung form (display all mode). You can also view the maximum amount of program logic on a screen by selecting a compressed rung mode.

The **Home**, **End**, **Page Up** and **Page Down**, **Previous**, **Next**, and cursor movement keys function the same, regardless of the view mode selected.

Display modes are selected from the View Modes Setup screen. To display this screen, press **View Mode** (**F5**) from the Programmer Setup menu.



```
|PROGRM |TABLES |STATUS |        |        |        |SETUP  |FOLDER |UTILTY |PRINT
1ports  2mode   3plcsel 4comset 5vumode 6        7        8        9        10
Setup data read from file: %LM090.PSU
>

          V I E W   M O D E S   S E T U P   (  A L T - N  )

            Program  View  Modes                              Enabled
        ________________________________________________    _______

        Nicknames                                                Y
        Reference Address Only                                   Y
        Reference Descriptions and Nicknames                     Y
        Reference Descriptions                                   Y
        Minimum Rung Size                                        N

            Reference   Table  View  Modes
        ________________________________________________

        Right to Left (Y),   Left to Right (N)                   Y


ID:          RUN/OUT EN    3ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                      PRG: LESSON
REPLACE
```

When the View Modes Setup screen is displayed, the mode selections which were last saved are shown. The default selections are **Y** (Yes) for the programmer view modes and **N** (No) for minimum rung size mode.

Use the Up and Down cursor keys to move between selection fields. Then, use the Tab key to toggle each selection between **Y** (Yes) and **N** (No), or enter **Y** for the view modes you want to sequence through when you press ALT-N. Enter **N** to disable the modes you do not want to sequence through.

To validate and save the view mode selections in the home directory file %LM090.PSU, press ALT-U or the **Escape** key. At least one selection must be **Y** (Yes) in order for the selections to be valid. If all view mode selections are **N** (No), an error message is displayed and the selections are not saved.

To abort the selections and use the last saved setup, press **ALT-A**.

Changing the View Mode

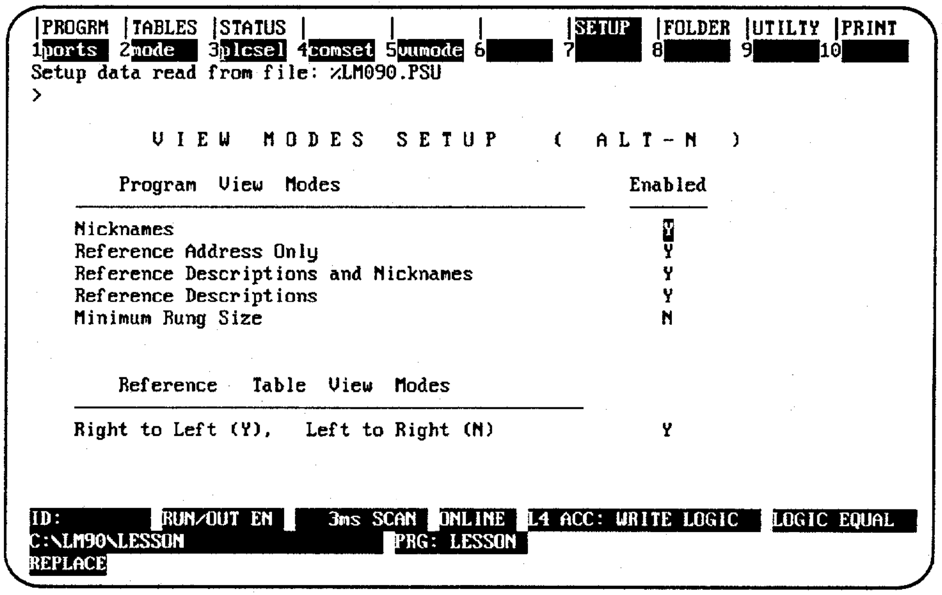
The **ALT-N** key sequence also enables you to display table data either right to left (lowest reference address on the right) or left to right (lowest reference address on the left). The view mode can be changed in any table (fixed or mixed) and in any program mode (**OFFLINE**, **ONLINE**, or **MONITOR**).

Note

The print function does not distinguish between these two display modes and prints all tables right to left.

The default view mode is the last view mode selected with **ALT-N**. If the view mode has never been changed, the default set in the programmer setup is used.

1. The default reference table view mode is displayed on the View Modes Setup screen. To display this screen, press **View Modes Setup** (**F5**) from the Programmer Setup menu. The view mode in the following screen is right to left (the default display).



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1ports  2mode   3plcsel 4comset 5vumode 6       7       8       9       10
Setup data read from file: %LM090.PSU
>

          V I E W   M O D E S   S E T U P    ( A L T - N )

         Program  View  Modes                          Enabled

      Nicknames                                           Y
      Reference Address Only                              Y
      Reference Descriptions and Nicknames                Y
      Reference Descriptions                              Y
      Minimum Rung Size                                   N


         Reference   Table  View  Modes

      Right to Left (Y),   Left to Right (N)              Y


ID:          RUN/OUT EN    3ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                      PRG: LESSON
REPLACE
```

2. To change the default view mode to display left to right, move the cursor to the *Enabled* field for the *Reference Table View Modes* entry. Then, use the Tab key to toggle the selection to **N** (No) or enter **N**.

3. To validate and save the default view mode selection in the home directory file %LM090.PSU, press **ALT-U** or the **Escape** key.

Section 6: Serial Printer Setup

The COM1 and COM2 serial ports can be used for serial printers. Beginning with Logicmaster 90-30/20 Release 4.50, the serial printer port ***must*** be configured with the MS-DOS mode command to match the printer settings.

Note

This is a significant change from the previous versions. If you are printing to a serial printer, make sure you observe the following guidelines:

- Either COM1 or COM2 can be used for serial printers; however, the serial printer port must be configured with the MS-DOS mode command to match the printer settings. The following printer settings must be configured: baud rate, parity, data bits, and stop bits. An example mode command for a serial printer operating on COM2 at 1200 baud, with 8 bits per character, 1 stop bit per character, and no parity bits would be:

```
mode com2:12,n,8,1
```

In the example mode command displayed above, *COM2:* is the port, *12* specifies 1200 baud, *n* specifies no parity, *8* specifies 8 bits per character, and *1* specifies 1 stop bit.

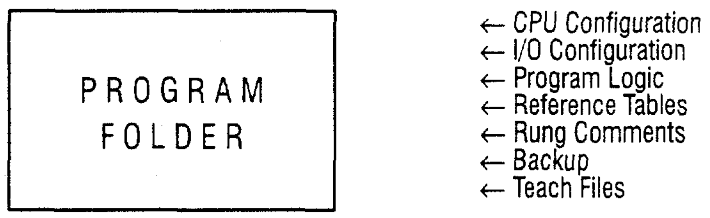
- The mode command can be entered as an MS-DOS command or placed in the AUTOEXEC.BAT file. The MS-DOS mode command must be used to configure the serial printer port before entering the Logicmaster 90-30/20/Micro software package.
- If persistent errors occur when printing listings to COM1 or COM2 from within the Logicmaster 90-30/20/Micro software package, the buffer size on the printer may have to be increased and the serial baud rate configured for the port may have to be lowered. Read the user manuals provided with your printer and the MS-DOS sections in the manual concerning serial ports, printing, and the mode command.

Note

The Standard Serial COM Port version of Logicmaster 90-30/20/Micro software requires either COM1, COM2, COM3, or COM4 as the communications port for communicating with the Series 90-30/20/Micro PLC. ***Do not use the same COM port*** for printing that you are using for communicating with the Series 90-30/20/Micro PLC.

Chapter 7 Program Folders

Series 90 programs and configuration data are stored in folders. A folder is an MS-DOS subdirectory which contains all information about one PLC program. To store more than one PLC program, you need to have one folder for **each** program. Folders must be created using Logicmaster 90-30/20/Micro software. An existing MS-DOS directory cannot be used as a folder.



Drawer

A drawer is an MS-DOS directory path which contains one or more folders.

Note

Drawers must be created using MS-DOS before they can be used by Logicmaster 90-30/20/Micro programming software.

When Logicmaster 90-30/20/Micro software is entered, the current MS-DOS directory is the default drawer. The default drawer may be changed from the Select screen by pressing **ALT-C** to clear the *Program Folder* field and then entering the MS-DOS directory path (ending in \) of the subdirectory containing the folders you want, for example:
C:\LM90\FOLDERS\.

Refer to appendix G for an explanation of the files created with Logicmaster 90-30/20/Micro software.

Program Folder Names

Every program will reside in its own program folder. The name of the program in the folder and the program folder will be the same (except for the "TEMP" folder, described below). The folder name is the program name in the PLC. The software will check these names to be sure they match. Programming functions, such as loading a program from the PLC to the computer, are not allowed if the names do not match.

TEMP Program Folder

The special program folder TEMP does not require the folder name and program name to match. The TEMP folder can be used if you need to view a user program and monitor its operation when you do not have the original folder on your machine.

The TEMP folder can also be used to load a copy of a program and modify it, without overwriting the previous version.

Note

Annotation files (nicknames, reference descriptions, and comment text) remain in the folder and are not stored to the PLC. Therefore, when loading a program from the PLC to the TEMP folder, the annotation will be missing. Those files are still in the original folder where the program was developed. If another program was developed in the TEMP folder prior to the download operation, the annotation in the recently loaded program will actually belong to the previous program.

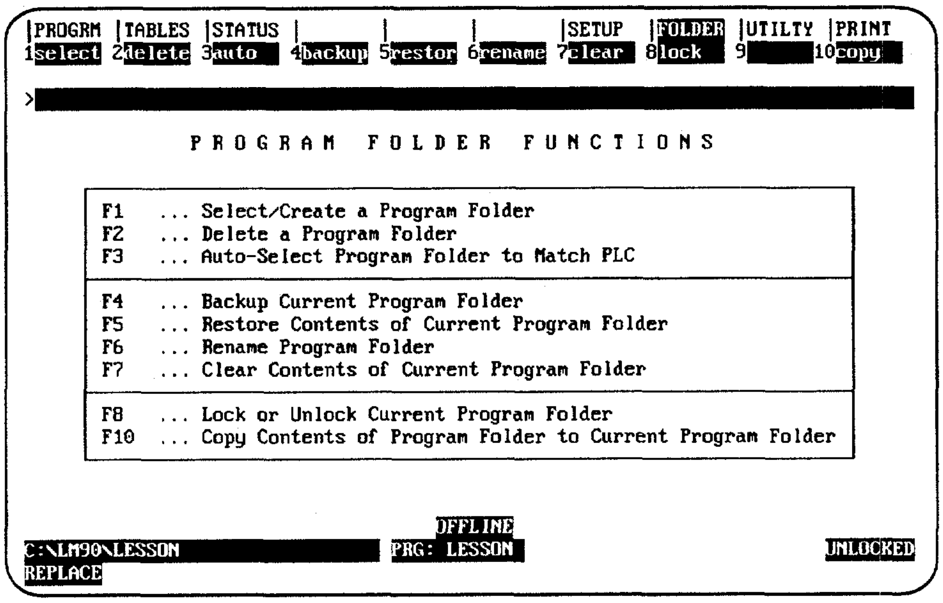
Using Program Folder Functions

Both the configuration software and programming software provide a group of program folder functions. These functions can be used to create, select, modify, or delete program folders.

Caution

Do not use MS-DOS to copy individual files from one folder to another or to delete files. Doing so may produce unexpected results. MS-DOS may only be safely used to copy an entire program folder to another program folder of the same name. If MS-DOS functions have been used to place program files into a program folder with a different name, you will not be able to select the program folder.

To use a program folder function, press **Folder** (**F8**) from the main menu.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1select 2delete 3auto   4backup 5restor 6rename 7clear  8lock   9       10copy
>

              P R O G R A M   F O L D E R   F U N C T I O N S

         F1   ... Select/Create a Program Folder
         F2   ... Delete a Program Folder
         F3   ... Auto-Select Program Folder to Match PLC

         F4   ... Backup Current Program Folder
         F5   ... Restore Contents of Current Program Folder
         F6   ... Rename Program Folder
         F7   ... Clear Contents of Current Program Folder

         F8   ... Lock or Unlock Current Program Folder
         F10  ... Copy Contents of Program Folder to Current Program Folder

                                        OFFLINE
C:\LM90\LESSON                          PRG: LESSON                    UNLOCKED
REPLACE
```

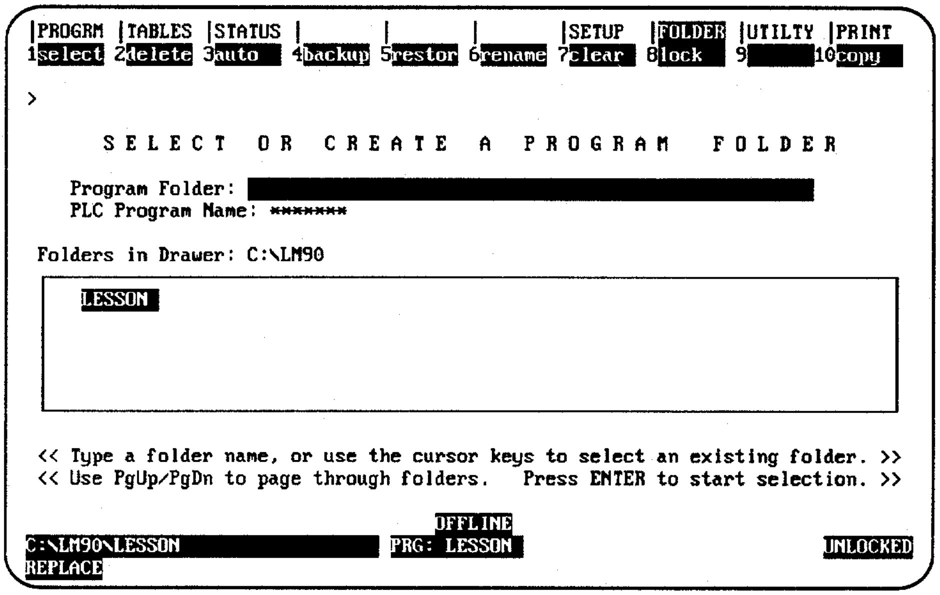
Function Key	Function	Description	Page
F1	Select	Create a new program folder, or select a previously created folder.	7-4
F2	Delete	Remove a program folder that is no longer needed.	7-7
F3	Auto	Automatically select the correct program folder and attempt to verify equality with the PLC program.	7-5
F4	Backup	Make a backup copy of the current program folder.	7-8
F5	Restore	Restore a program folder with its backup copy.	7-9
F6	Rename	Rename a program folder to a new folder name.	7-11
F7	Clear	Clear the contents of the current program folder.	7-12
F8	Lock	Lock or unlock a program folder. Locking a program folder prevents its files from being changed or deleted.	7-13
F10	Copy	Copy a program folder into the current program folder. Use this function to make a copy of a program folder that can be modified while retaining the original.	7-14

Note

If a folder contains locked subroutines, these blocks remain locked when the Logicmaster 90-30/20/Micro software copy, backup, and restore folder functions are used. For more information on locking and unlocking subroutines, refer to chapter 3, section 8, "Subroutine Blocks."

Selecting/Creating Program Folders

To create a new program folder or use one that already exists, press **Select** (**F1**) from the Program Folder Functions menu.



```
|PROGRM |TABLES |STATUS |      |      |      |SETUP  |FOLDER |UTILTY |PRINT
1select 2delete 3auto   4backup 5restor 6rename 7clear  8lock   9       10copy

>

     S E L E C T   O R   C R E A T E   A   P R O G R A M   F O L D E R

   Program Folder:
   PLC Program Name: ********

 Folders in Drawer: C:\LM90

    LESSON

 << Type a folder name, or use the cursor keys to select an existing folder. >>
 << Use PgUp/PgDn to page through folders.   Press ENTER to start selection. >>

                                          OFFLINE
C:\LM90\LESSON                           PRG: LESSON                UNLOCKED
REPLACE
```

Note

Once you edit a program folder using Release 3 or higher of Logicmaster 90-30/20 software, you cannot use this folder with an earlier release of software.

Selecting a Previously Created Folder with a New 351 Configuration

If you are using a 351 CPU hardware configuration and select a folder created previously when you had a different CPU configuration, you will be prompted to convert the folder to one that uses the code written for 351 CPUs.

Note

As mentioned above, Logicmaster prompts you if you are using 351 hardware configuration and then select a folder created under a different CPU configuration. If you plan to use that folder again with lower model CPUs, Logicmaster does allow that; but make sure you stay within the memory limit and program size limits of the lower model CPU.

The name of the last folder selected is shown in reverse video in the *Program Folder* field. Other program folders in the current drawer are also listed on the screen. If the attached PLC contains a user program, the name of this program appears in the *PLC Program Name* field. If the attached PLC does not contain a user program, this field is blank. If the programmer is offline or not connected to a PLC, this field contains asterisks. An entry in this field may not be edited.

To select a folder in the current drawer, move the cursor to the desired folder name or type the name of the program folder in the *Program Folder* field, and press the **Enter** key. (This is also the name of the program.) The name can have up to seven characters. If a second floppy disk is inserted after a folder has already been selected or created on the first floppy disk, the list of folders is not updated. In order to have the folder list updated, you must reselect the drawer (e.g., enter **A:\**).

To select a folder in a different drawer, first select the drawer by entering the drawer MS-DOS path, ending with a "\", and pressing the **Enter key. For example, enter C:\LM90\FOLDERS\**. When the **Enter** key is pressed, the names of the folders in the newly selected drawer are displayed. Position the cursor on the name of the program folder you wish to select or type the name of the program folder, and press the **Enter** key.

If the program folder already exists, select whether or not to create a backup copy. If the program folder does not already exist, the software will prompt you for confirmation to create a new program folder. This prevents you from accidentally creating a program folder due to a typing mistake. To cancel any changes made to this screen, press **ALT-A** (abort). To return to the Program Folder Functions menu, use the **Escape** key.

Auto-Select Function

The **Auto** (**F3**) softkey is only active when the select screen is displayed. When **F3** is pressed in **OFFLINE** mode or if the programmer is not communicating with the PLC, a message is displayed indicating that the auto-select function is not available in that mode.

When **F3** is pressed in **MONITOR** or **ONLINE** mode, the programming software checks to see if a folder exists under the current directory whose name matches the name of the program in the PLC. If the folder does exist, it is automatically selected. The software then attempts to verify the program or configuration in the folder with that in the PLC, and updates the *equality* field on the status line based on the result. After the select screen is exited, the main menu is displayed.

In the programming software, if a matching folder does not exist, the following message is displayed: "Folder matching PLC name not found; load program to TEMP folder? (Y/N)". If **N** (No) is entered, no folder is selected and the select screen remains displayed on the screen.

If **Y** (Yes) is entered, the TEMP folder is selected or created if one does not exist. Program logic and configuration are loaded from the PLC into the TEMP folder. If the TEMP folder already contains logic and/or configuration files, those files are backed up prior to the load. The status lines are updated to show that the folder program is equal to the version in the PLC.

In the configuration software, the message displayed when a matching folder does not exist is "load configuration to TEMP folder?".

If an error is encountered verifying or loading data from the PLC, an error message is displayed and the select screen continues to be displayed.

Automatic Folder Selection

The automatic folder selection feature allows Logicmaster 90-30/20/Micro software to automatically select the correct program folder and attempt to verify equality with the PLC program.

During power-up initialization, when the initializing screen is displayed, Logicmaster 90-30/20/Micro software attempts to establish point-to-point communications with an attached PLC. If this attempt is not successful, the initial folder selection screen is displayed. The programmer mode will default to **OFFLINE** mode on computers without a programmer mode keyswitch.

If a connection exists, computers without a keyswitch default to **MONITOR** mode. Logicmaster 90-30/20/Micro software will then check to see if a folder whose name matches the PLC program name exists. If the folder does exist, that folder is automatically selected and an equality check is performed. The main menu screen is displayed after the equality check is completed.

If a matching folder does not exist, the initial folder selection screen is displayed. You can select a folder or press **Auto** (**F3**) to load to the TEMP folder, or select a new drawer and press **Auto** (**F3**) to try again to select the matching folder and then to check to see if they are both the same.

Note

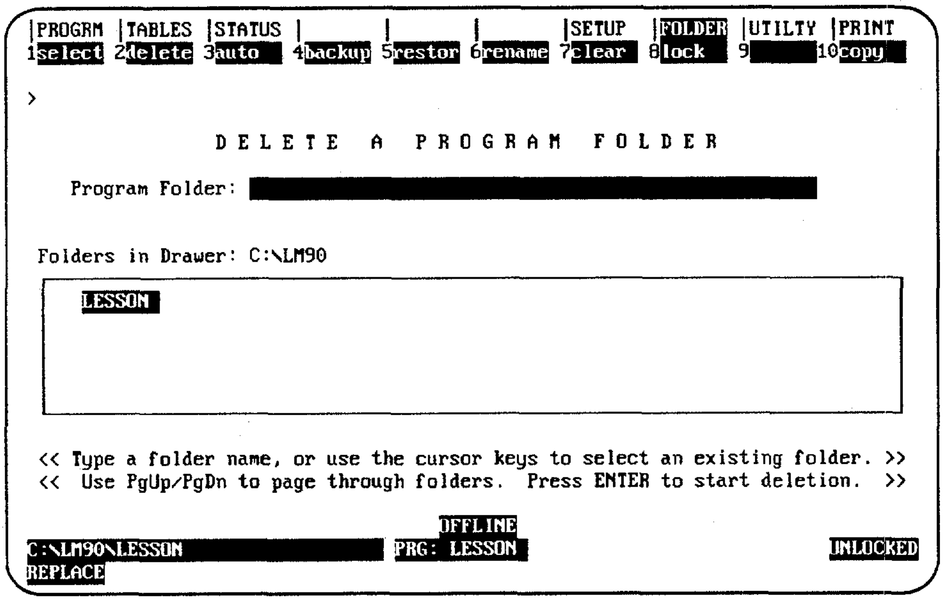
If the baud rate with the serial version of Logicmaster 90-30/20/Micro software is set to 1200 or less, the programmer does not automatically go to **MONITOR** mode.

Deleting Program Folders

Use this function to remove a program folder that is no longer needed. If the program folder has a backup, that copy is also deleted automatically. If you have created any directories using MS-DOS beneath the program folder to be deleted, you must remove them before using the delete function. You may not be able to delete a program folder residing on a virtual drive.

If a program folder is locked (lock status is shown in the lower right corner of your screen), you cannot delete it. Please refer to the information on unlocking program folders provided later in this chapter.

Press **Delete** (**F2**) from the Program Folder Functions menu.


```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1select 2delete 3auto   4backup 5restor 6rename 7clear  8lock   9       10copy

>

              D E L E T E   A   P R O G R A M   F O L D E R

     Program Folder:

  Folders in Drawer: C:\LM90

     LESSON

 << Type a folder name, or use the cursor keys to select an existing folder. >>
 <<  Use PgUp/PgDn to page through folders.  Press ENTER to start deletion.  >>

                                    OFFLINE
C:\LM90\LESSON                      PRG: LESSON                        UNLOCKED
REPLACE
```

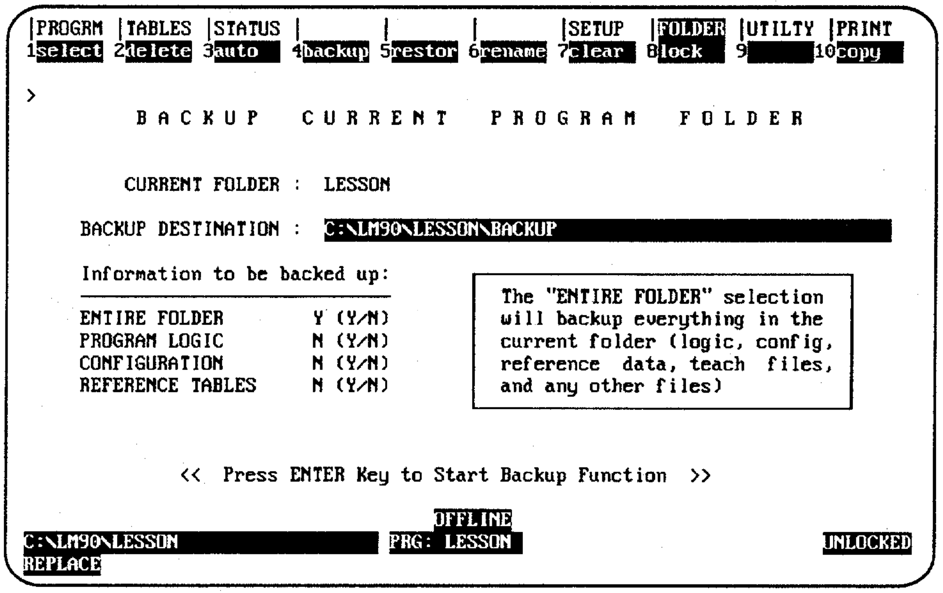
To delete a folder in the current drawer, type the name of the program folder in the *Program Folder* field or move the cursor to the desired folder name. Then, press the **Enter** key. The software will not allow the currently selected folder to be deleted.

To delete a folder in a different drawer, enter the full path specification or specify the drawer in the *Program Folder* field. The window is changed to display the folders in this drawer.

Respond to the confirmation prompt to continue with the deletion. *Once started, the delete operation cannot be aborted.* To return to the Program Folder Functions menu, press the **Escape** key.

Backing Up Program Folders

To create a backup copy of the currently selected program folder, press **Backup** (**F4**) from the Program Folder Functions menu.


```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1select 2delete 3auto   4backup 5restor 6rename 7clear  8lock   9       10copy
>
        B A C K U P    C U R R E N T    P R O G R A M    F O L D E R

           CURRENT FOLDER :  LESSON

       BACKUP DESTINATION :  C:\LM90\LESSON\BACKUP

       Information to be backed up:
       ----------------------------          The "ENTIRE FOLDER" selection
       ENTIRE FOLDER       Y (Y/N)           will backup everything in the
       PROGRAM LOGIC       N (Y/N)           current folder (logic, config,
       CONFIGURATION       N (Y/N)           reference  data,  teach  files,
       REFERENCE TABLES    N (Y/N)           and any other files)

              << Press ENTER Key to Start Backup Function >>

                                        OFFLINE
C:\LM90\LESSON                          PRG: LESSON                    UNLOCKED
REPLACE
```

The backup folder is located in a subdirectory under the program folder. If no backup folder exists for the current program folder, one is created automatically.

The *Backup Destination* field allows you to specify where you want to put the backup archive. The default destination is the backup directory of the current folder. To change the destination, move the cursor to this field and enter the new destination.

Note

When backing up to a floppy diskette, you must back up to a subdirectory. For example, **A:\** cannot be used as the backup destination. If **LESSON** is the folder name, **A:\LESSON** should be used. If only **A:\** is specified, you cannot restore from the floppy diskette.

Use the cursor keys to move from one option field to another. To back up the entire contents of the current folder, enter **Y** (Yes) in the *Entire Folder* field. To back up only selected options, enter **Y** (Yes) in the corresponding fields:

Field	Description
Program Logic	The ladder logic program.
Configuration	The current configuration.
Reference Tables	The reference tables for the program.

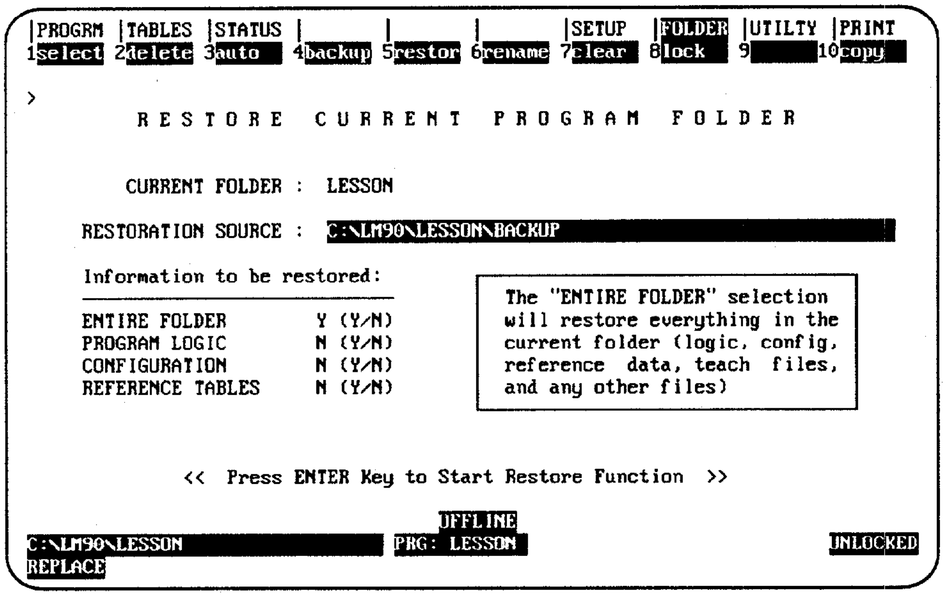
Then press the **Enter** key. If you back up the entire folder, the software will prompt you for confirmation and then automatically delete the previous backup. A new backup archive will be created to reflect the current contents of the program folder. If you back up only program logic or any other options besides the entire folder, the previous backup is left intact and the selected options are added to the backup. If the selected options were already stored to the backup, they will first be deleted.

Once started, the backup operation cannot be aborted. To return to the Program Folder Functions menu, press the **Escape** key.

Restoring Program Folders

To replace the contents of the current program folder with the backup copy, press **Restore** (**F5**) from the Program Folder Functions menu. To restore a program folder, a backup copy must exist.

If the program folder is locked (locked status is shown in the lower right corner of your screen), you must change the status to unlocked before a restore can occur. Please refer to the information on unlocking program folders provided later in this chapter.



```
|PROGRM |TABLES |STATUS |        |        |        |SETUP  |FOLDER |UTILTY |PRINT
1select 2delete 3auto   4backup  5restor  6rename  7clear  8lock   9       10copy
>
           R E S T O R E   C U R R E N T   P R O G R A M   F O L D E R

              CURRENT FOLDER  :  LESSON

          RESTORATION SOURCE  :  C:\LM90\LESSON\BACKUP

          Information to be restored:
          ____________________________     ┌──────────────────────────────────┐
                                           │ The "ENTIRE FOLDER" selection    │
          ENTIRE FOLDER       Y (Y/N)      │ will restore everything in the   │
          PROGRAM LOGIC       N (Y/N)      │ current folder (logic, config,   │
          CONFIGURATION       N (Y/N)      │ reference  data, teach  files,   │
          REFERENCE TABLES    N (Y/N)      │ and any other files)             │
                                           └──────────────────────────────────┘

                 << Press ENTER Key to Start Restore Function >>

                                          OFFLINE
C:\LM90\LESSON                            PRG: LESSON                  UNLOCKED
REPLACE
```

The *Restoration Source* field allows you to specify where you want to retrieve the backup archive from. The default source is the backup directory of the current folder. To change the source, move the cursor to this field and enter the new source.

Note

The backup archive must be restored to a folder with the same name as the folder from which the backup originally was made. If not, the message "No backup found to restore" is displayed.

Use the cursor keys to move from one option field to another. To restore the entire contents of the current folder, enter **Y** (Yes) in the ***Entire Folder*** field. To restore only selected options, enter **Y** (Yes) in the corresponding fields:

Field	Description
Program Logic	The ladder logic program.
Configuration	The current configuration.
Reference Tables	The reference tables for the program.

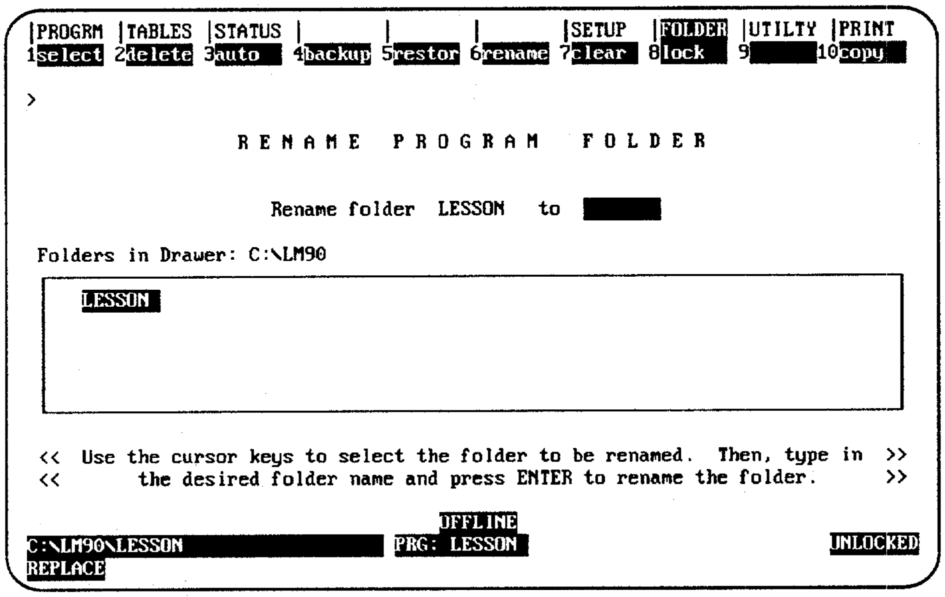
Then press the **Enter** key. If any information exists in the current program folder that would be written over by the backup copy, the software will prompt you for confirmation. *Once started, the restore operation cannot be aborted.*

If the programmer is in **ONLINE** or **MONITOR** mode, an auto-verification is performed to determine whether the restored program or configuration is equal to the PLC.

To return to the Program Folder Functions menu, press the **Escape** key.

Renaming Program Folders

This feature of Logicmaster 90-30/20/Micro software enables you to rename any folder in the current drawer to another folder name that is not being used. To rename a program folder, press **Rename** (**F6**) from the Program Folder Functions menu or from another Program Folder screen.



```
|PROGRM |TABLES |STATUS |        |        |        |SETUP  |FOLDER |UTILTY |PRINT
1select 2delete 3auto   4backup  5restor  6rename  7clear  8lock   9       10copy

>

              R E N A M E   P R O G R A M   F O L D E R

                    Rename folder  LESSON   to

 Folders in Drawer: C:\LM90

    LESSON

 <<  Use the cursor keys to select the folder to be renamed.  Then, type in  >>
 <<       the desired folder name and press ENTER to rename the folder.      >>

                                       OFFLINE
C:\LM90\LESSON                        PRG: LESSON                     UNLOCKED
REPLACE
```

Enter the new folder name into the *Rename Folder to* field. If you press an alphanumeric key, it is automatically displayed in the *Rename Folder to* field. To clear this field, press **ALT-C** or use the **Delete** key to delete the contents of the field.

Press the **Enter** key to start the renaming operation. The software will prompt you for confirmation to continue the operation with the message "Current folder (<file_name>) will be renamed to <file_name>. Continue? (Y/N)." *Once started, the rename operation cannot be aborted.*

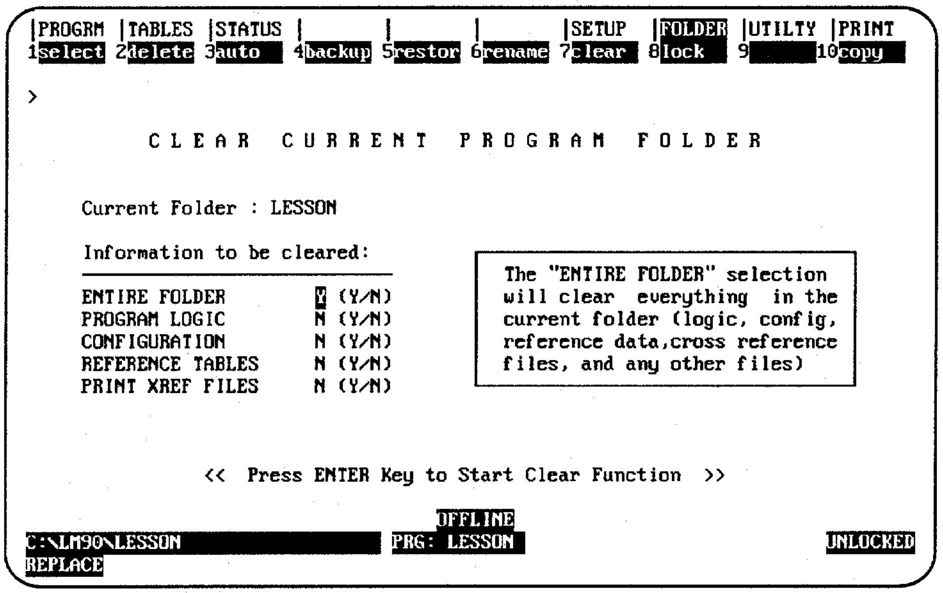
Once the rename operation is complete, the message "Folder renamed successfully" is displayed and the *Rename Folder to* field is cleared.

If the rename operation is aborted, the *Rename Folder to* field is not automatically cleared since the rename never completed. You must manually clear the field by pressing **ALT-C** or use the **Delete** key to delete the contents of the field.

Clearing Program Folders

To delete the contents of the current program folder while keeping the folder itself for future use, press **Clear** (**F7**) from the Program Folder Functions menu.

If a program folder is locked (look at the lower right corner of your screen), you cannot clear it. Please refer to the information on unlocking program folders provided later in this chapter.



```
|PROGRM |TABLES |STATUS |        |        |        |SETUP  |FOLDER |UTILTY |PRINT
1select 2delete 3auto   4backup 5restor 6rename 7clear  8lock   9       10copy

>

        C L E A R   C U R R E N T   P R O G R A M   F O L D E R

    Current Folder : LESSON

    Information to be cleared:
    ----------------------------------         The "ENTIRE FOLDER" selection
    ENTIRE FOLDER         Y (Y/N)              will clear  everything  in the
    PROGRAM LOGIC         N (Y/N)              current folder (logic, config,
    CONFIGURATION         N (Y/N)              reference data,cross reference
    REFERENCE TABLES      N (Y/N)              files, and any other files)
    PRINT XREF FILES      N (Y/N)

            << Press ENTER Key to Start Clear Function >>

                                          OFFLINE
C:\LM90\LESSON                          PRG: LESSON                      UNLOCKED
REPLACE
```

Use the cursor keys to move from one option field to another. To clear the entire contents of the current folder, enter **Y** (Yes) in the *Entire Folder* field. To clear only selected options, enter **Y** (Yes) in the corresponding fields:

Field	Description
Program Logic	The ladder logic program.
Configuration	The current configuration.
Reference Tables	The reference tables for the program.
Print Cross Reference Files	The PRINT.XOV and all .XRF files for the program.

The Tab key may also be used to toggle the selection of each option. The default selection for the *Entire Folder* field is **Y** (Yes); all other fields default to **N** (No). Once the *Entire Folder* field is set to **Y**, the remaining fields are automatically set to **N**. Once any of the other fields is set to **Y**, the *Entire Folder* field is automatically set to **N**.

Note

Program annotation files only exist in the folder and not in the PLC. Clearing the folder and then loading the program from the PLC results in lost annotation.

Press the **Enter** key to begin the clear operation. If there is any information in the folder, the software will prompt you for confirmation to continue the clear operation. *Once started, the clear operation cannot be aborted.*

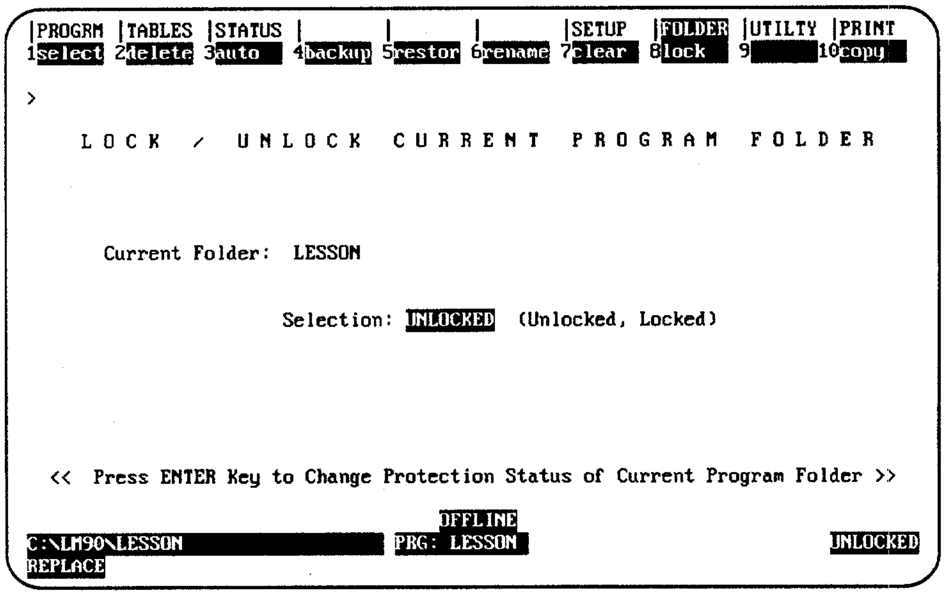
If you answer the continuation prompt by pressing **Y** (Yes), you will have a chance to back up the current program folder. To create a backup copy, press **Y** (Yes) at the prompt. Press **N** (No) if you do not want to back up the information first.

If the programmer is in **ONLINE** or **MONITOR** mode, the status is automatically set to **NOT EQUAL**.

To return to the Program Folder Functions menu, press the **Escape** key.

Locking/Unlocking Program Folders

Locking a program folder protects its files from being accidentally changed. To change the locked status of the current program folder, press **Lock** (**F8**) from the Program Folder Functions menu.



```
|PROGRM |TABLES |STATUS |        |        |        |SETUP  |FOLDER |UTILTY |PRINT
1select 2delete 3auto   4backup 5restor 6rename 7clear  8lock   9       10copy

>

   L O C K  /  U N L O C K   C U R R E N T   P R O G R A M   F O L D E R


     Current Folder:  LESSON

                    Selection: UNLOCKED  (Unlocked, Locked)



  << Press ENTER Key to Change Protection Status of Current Program Folder >>
                                          OFFLINE
C:\LM90\LESSON                           PRG: LESSON                UNLOCKED
REPLACE
```

Use the Tab key to change the access description. Then press the **Enter** key. The new access description will appear in the lower right corner of the screen.

Note

Program folders on write-protected floppy diskettes are automatically locked. Remove the write-protect tab and unlock the folder using this function.

Copying Program Folders

Copying to Another Folder on the Hard Disk

Use the copy function to copy from another program folder into the current folder. The source program folder is copied into the "Current" program folder. If there are any files in the current folder, they are destroyed by the copy process.

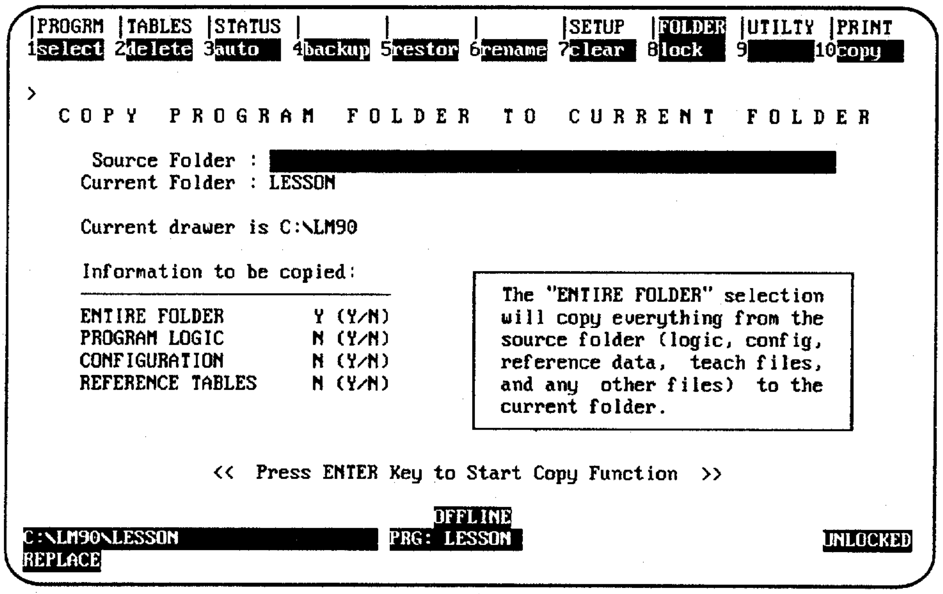
Note

The Copy feature functions somewhat differently than you might expect. Notice that you are copying another program folder *into* the current folder. That is, if you have been using a folder called "LESSON," then LESSON is the name of your current folder (as displayed in the Status line toward the bottom of your screen). If you copy the folder called "ACCTRL," then the folder called LESSON becomes an exact copy of ACCTRL.

Note

The copy function is only available in the programming software; it is not available in the configuration software.

To make a copy of a program folder, press **Copy** (**F10**) from the Program Folder Functions menu.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1select 2delete 3auto   4backup 5restor 6rename 7clear  8lock   9       10copy
>
   C O P Y   P R O G R A M   F O L D E R   T O   C U R R E N T   F O L D E R

     Source Folder  :
    Current Folder  : LESSON

    Current drawer is C:\LM90

    Information to be copied:
    ________________________
    ENTIRE FOLDER        Y (Y/N)       The "ENTIRE FOLDER" selection
    PROGRAM LOGIC        N (Y/N)       will copy everything from the
    CONFIGURATION        N (Y/N)       source folder (logic, config,
    REFERENCE TABLES     N (Y/N)       reference data, teach files,
                                       and any other files) to the
                                       current folder.

              << Press ENTER Key to Start Copy Function >>
                                       OFFLINE
C:\LM90\LESSON                         PRG: LESSON                    UNLOCKED
REPLACE
```

Enter the name of the program folder whose contents you want to copy into the currently selected program folder. Then, use the cursor keys to move from one option field to another. To copy the entire contents of the source folder, enter **Y** (Yes) in the

Entire Folder field. To copy only selected options, enter **Y** (Yes) in the corresponding fields:

Field	Description
Program Logic	The ladder logic program.
Configuration	The current configuration.
Reference Tables	The reference tables for the program.

The Tab key may also be used to toggle the selection of each option. The default selection for the *Entire Folder* field is **Y** (Yes); all other fields default to **N** (No). Once the *Entire Folder* field is set to **Y**, the remaining fields are automatically set to **N**. Once any of the other fields is set to **Y**, the *Entire Folder* field is automatically set to **N**.

Press the **Enter** key to begin the copy operation. If there is any information in the current folder, the software will prompt you for a confirmation to continue the copy operation since everything in the current folder will be overwritten. *Once started, the copy operation cannot be aborted.*

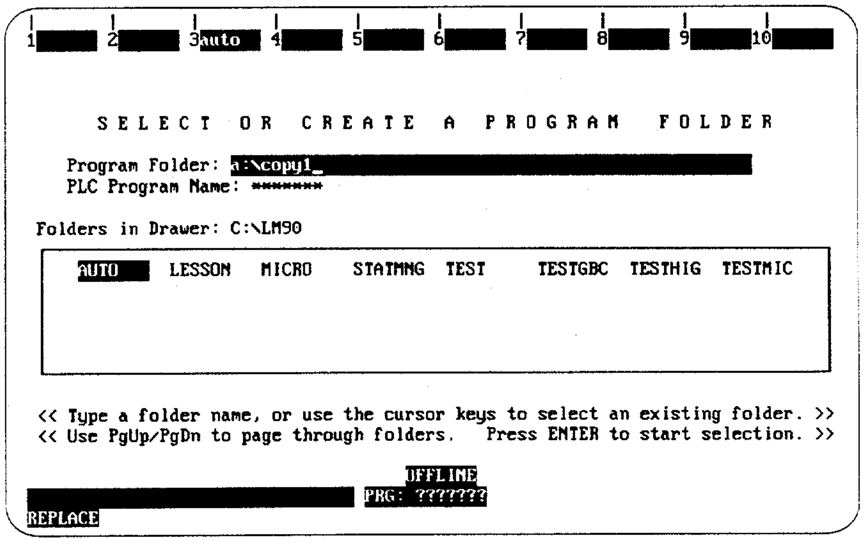
If the programmer is in **ONLINE** or **MONITOR** mode, an auto-verification is performed to determine whether the programmer/configurator is equal to the PLC.

To return to the Program Folder Functions menu, press the **Escape** key.

Copying to a Diskette

To copy to a diskette, you will need to follow these steps:

1. Place a diskette in your disk drive (usually an A or B drive). There should be plenty of room on the diskette.
2. Press **F8** from the main menu to select the **Folder** functions.
3. Press **select** (**F1**) "Select/Create a Program Folder" (see page 7-3 for screen sample if needed).
4. In the Program Folder blank, type in the name you wish to use for the copy you are about to make. Make sure you specify the disk drive letter, e.g., a:\ as shown below:



```
1     2     3auto  4     5     6     7     8     9     10

     S E L E C T   O R   C R E A T E   A   P R O G R A M   F O L D E R

  Program Folder: a:\copy1_
  PLC Program Name: ********

Folders in Drawer: C:\LM90

  AUTO   LESSON   MICRO   STATMNG   TEST   TESTGBC   TESTHIG   TESTMIC

<< Type a folder name, or use the cursor keys to select an existing folder. >>
<< Use PgUp/PgDn to page through folders.   Press ENTER to start selection. >>

                                     OFFLINE
                                     PRG: ???????
REPLACE
```

5. Press **Enter** to select this new folder on your A or B drive.

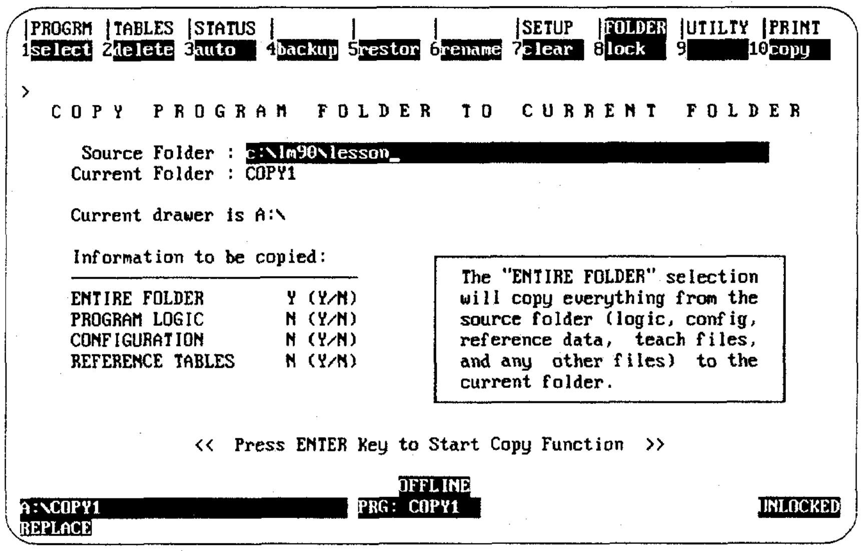
The following prompt will appear at the top of your screen: "Program folder does not exist; create new folder? (Y/N)"

6. Type **Y** to confirm that you want to create a new folder on your diskette.

Note

Notice that you are only creating an empty folder. In the next steps you will copy the contents of your source folder into that empty folder.

7. After Logicmaster has finished creating a new folder, press the **Copy** softkey (**F10**).



```
 |PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1select 2delete 3auto   4backup 5restor 6rename 7clear  8lock   9       10copy

>
   C O P Y   P R O G R A M   F O L D E R   T O   C U R R E N T   F O L D E R

     Source Folder : c:\lm90\lesson_
    Current Folder : COPY1

    Current drawer is A:\

    Information to be copied:
    ___________________________
    ENTIRE FOLDER        Y (Y/N)     The "ENTIRE FOLDER" selection
    PROGRAM LOGIC        N (Y/N)     will copy everything from the
    CONFIGURATION        N (Y/N)     source folder (logic, config,
    REFERENCE TABLES     N (Y/N)     reference data,  teach files,
                                     and any  other files)  to the
                                     current folder.

             << Press ENTER Key to Start Copy Function >>

                                       OFFLINE
A:\COPY1                               PRG: COPY1                      UNLOCKED
REPLACE
```

8. Type in the entire path and folder name, e.g., **c:\lm90\lesson** as shown above. Be sure to include the DOS drive and directory which is usually c:\lm90.

9. Press **Enter** to start the Copy function.

When the Copy is complete, the words "Selected folder items have been copied to current folder" will appear towards the top of your screen.

10. Press the **Escape** key to exit the Copy function which takes you to the Program Folder Functions. Press the **Escape** key again to return to the Program Folder Functions menu. Press the **Escape** key again if you wish to exit the Programming software.

    (An alternate way of exiting the software is to press **Ctrl-Break** once. A prompt will appear asking you if you wish to exit. Type **Y** (for Yes) to return to the Logicmaster main menu where you can press either **F10** or the **Escape** key to exit. When exiting through **Ctrl-Break**, you do not need to press the **Escape** key multiple times.)

You can now remove your diskette which has a copy of the source folder on it.

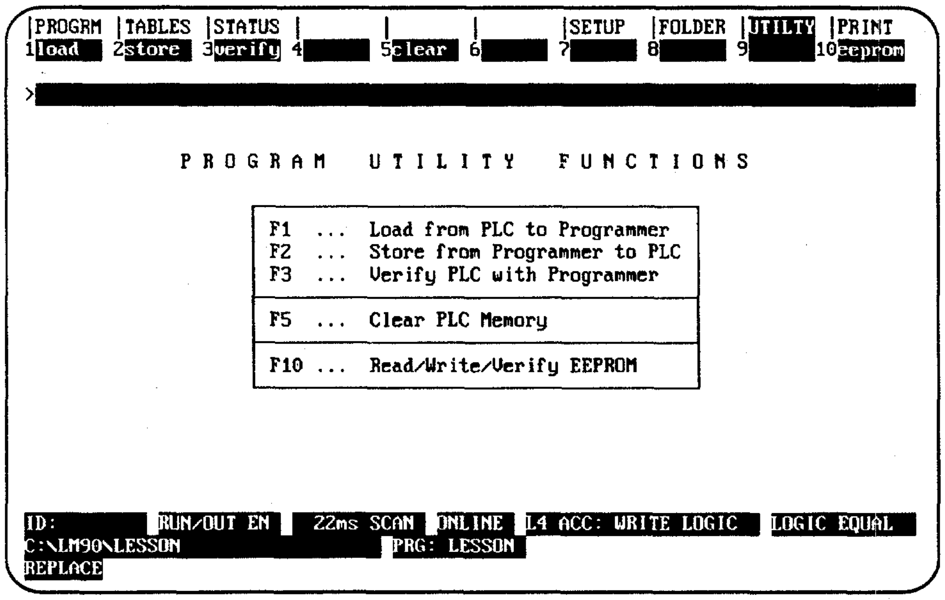
Chapter 8 Program Utilities

The program utility functions are used to transfer programs, configuration data, and reference tables between the programmer and the PLC. They are also used to compare the program, configuration data, and reference tables in the programmer with the program, configuration data, and reference tables in the PLC, to clear PLC memory, and to read/write/verify EEPROM.

Note

In the configuration software, only the configuration may be loaded, stored, verified, or cleared. No operations on program logic or tables may be performed.

To access the program utility functions, press **Utility** (**F9**) from the main menu, or **Shift-F9** from any main menu function screen.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1load   2store  3verify 4       5clear  6       7       8       9       10eeprom
>

                 P R O G R A M   U T I L I T Y   F U N C T I O N S

                       F1  ...  Load from PLC to Programmer
                       F2  ...  Store from Programmer to PLC
                       F3  ...  Verify PLC with Programmer

                       F5  ...  Clear PLC Memory

                       F10 ...  Read/Write/Verify EEPROM

ID:            RUN/OUT EN   22ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                         PRG: LESSON
REPLACE
```

Function Key	Function	Description	Page
F1	Load	Copy program logic, configuration data, and/or reference tables from the PLC to the programmer.	8-3
F2	Store	Copy program logic, configuration data, and/or reference tables from the programmer to the PLC.	8-5
F3	Verify	Compare program logic, configuration data, and/or reference data in the programmer with the program logic, configuration data, and/or reference tables in the PLC.	8-8
F5	Clear	Delete program, configuration data, and/or reference tables from PLC memory.	8-10
F10	EEPROM	Read the EEPROM contents into PLC memory, write the entire contents of PLC memory to EEPROM, or verify the EEPROM contents with the PLC memory.	8-12

Loading from PLC to Programmer

Use the load function to transfer program logic, configuration data, and/or reference tables from a PLC to the programmer. The load function transfers the program, which remains unchanged in the PLC. This function may be password protected in the PLC. If so, you must know the password in order to use the function.

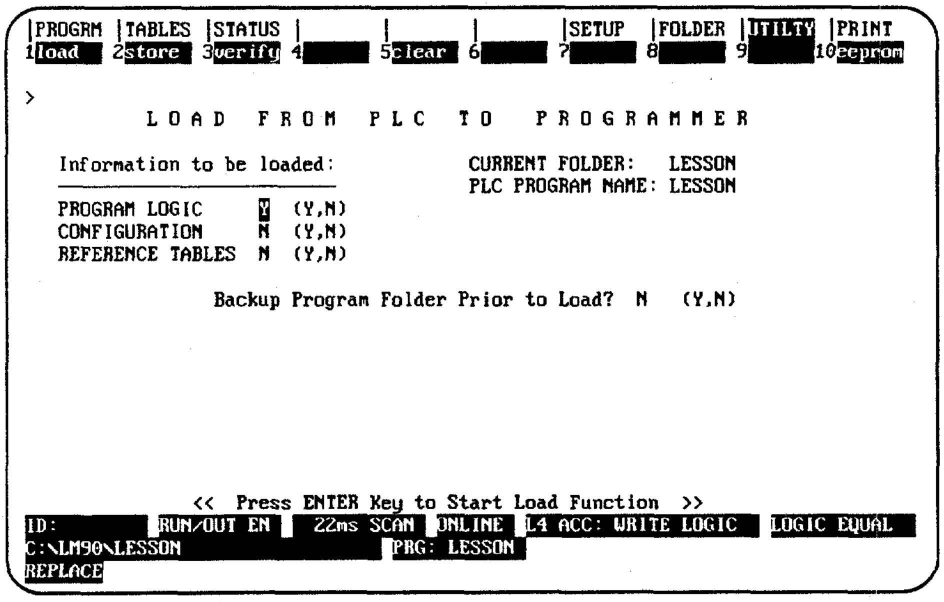
Caution

You cannot load a Release 4 or above Program with a release earlier than 3.50 of of Logicmaster 90-30/20 software.

Note

In the configuration software, only the configuration may be loaded. No operations on program logic or tables may be performed.

To use the load function, the programmer must be in `ONLINE` or `MONITOR` mode. Press `Load` (`F1`) from the Program Utility Functions menu or from another Program Utilities screen. The Load Program screen appears:



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1load   2store  3verify 4       5clear  6       7       8       9       10eeprom

>
          L O A D   F R O M   P L C   T O   P R O G R A M M E R

  Information to be loaded:                 CURRENT FOLDER:   LESSON
  ________________________                  PLC PROGRAM NAME: LESSON
  PROGRAM LOGIC      Y  (Y,N)
  CONFIGURATION      N  (Y,N)
  REFERENCE TABLES   N  (Y,N)

                 Backup Program Folder Prior to Load?  N    (Y,N)




                 << Press ENTER Key to Start Load Function >>
ID:               RUN/OUT EN   22ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                             PRG: LESSON
REPLACE
```

Note

Annotation files (nicknames, reference descriptions, and comment text) remain in the folder and are not stored to the PLC. Therefore, you must load from the PLC to the original folder containing the annotation in order to retain its annotation.

Note

If comment text has been edited since the program was stored to the PLC, do not load the program to the folder to restore equality. The program with the updated text should be stored to the PLC instead. This also applies to identifier descriptions for subroutine blocks which have been edited off-line. The subroutine block containing the edited identifier descriptions should be stored to the PLC in order to restore equality.

The Load Program screen shows the currently selected program folder and the program name in the PLC; these cannot be changed on this screen. A program must be loaded to a folder whose name matches the PLC program name or a TEMP folder (e.g., a folder whose name is TEMP).

Three types of data can be loaded from the PLC to the programmer: program logic, configuration data, or reference tables. When this screen first appears, only the program logic is set to **Y** (Yes). To load all of the data, change the selections for reference tables and configuration to **Y** (Yes). To load only part of the data, select **N** (No) for any of the following:

Field	Description
Program Logic	The ladder logic program.
Configuration	The current configuration.
Reference Tables	The reference tables for the program.

Also, select whether or not you want to create a backup copy of the current program folder before loading the data. Use the cursor keys to select items, and type in new selections as appropriate. To restore the original selections while editing this screen, press **ALT-A**.

To begin loading the data, press the **Enter** key.

The name of the program in the PLC is checked against the name of the current program folder. If the names are the same, the load continues. If the names are not the same, you cannot start the load unless the folder name is TEMP.

After a successful transfer of data, the software displays the message "Load Complete". If a communication error occurs during the load process (indicated by a message on the screen), the selected items are cleared from the current folder. Correct the error and repeat the load function.

To stop a program transfer in progress, press **ALT-A**.

To return to the Program Utility Functions menu, press the **Escape** key.

Storing to PLC from Programmer

Use the store function to copy program logic, configuration data, and/or reference tables from the programmer to the PLC. The store function copies the program, which remains unchanged in the programmer. If the PLC program name is not the same as the folder name, the store function clears both the program and configuration data from the PLC. The data is then stored from the new program folder.

The store function may be password protected in the PLC. If so, you must know the password in order to use the function.

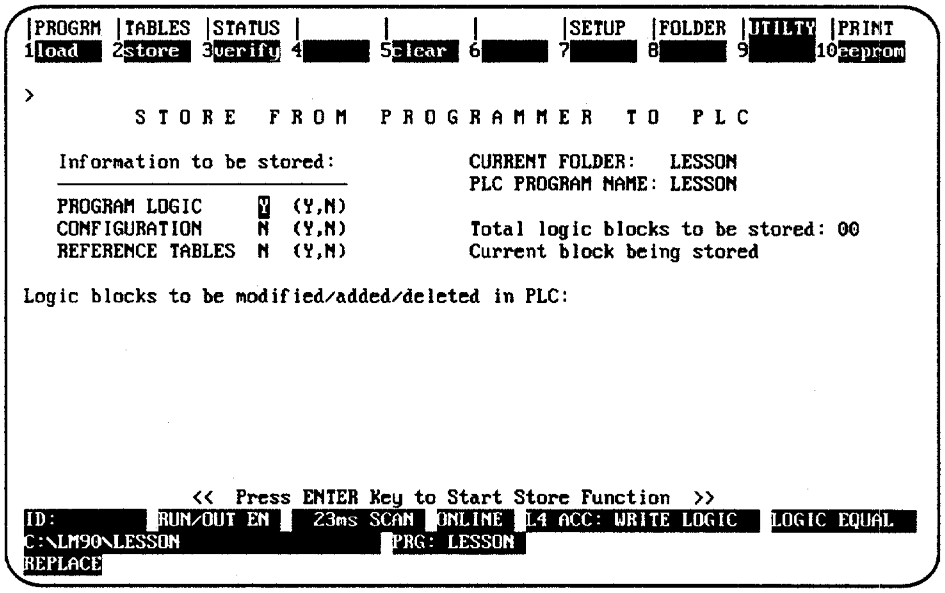
Note

In the configuration software, only the configuration may be stored. No operations on program logic or tables may be performed.

To use the store function, the programmer must be in **ONLINE** mode. The PLC can be in either **STOP** or **RUN** mode. (For more information, see "Run Mode Store Function" on page 8-6.).

Logic Size (Bytes)	2400 Baud	19.2K Baud
1K	12 seconds	2 seconds
2K	17 seconds	3 seconds
4K	26 seconds	4 seconds
8K	45 seconds	8 seconds
16K	84 seconds	13 seconds

Press **Store** (**F2**) from the Program Utility Functions menu. The Store Program screen appears:


```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1load   2store  3verify 4       5clear  6       7       8       9       10eeprom

>
             S T O R E   F R O M   P R O G R A M M E R   T O   P L C

    Information to be stored:                 CURRENT FOLDER:    LESSON
                                              PLC PROGRAM NAME:  LESSON
    PROGRAM LOGIC      Y  (Y,N)
    CONFIGURATION      N  (Y,N)               Total logic blocks to be stored: 00
    REFERENCE TABLES   N  (Y,N)               Current block being stored

Logic blocks to be modified/added/deleted in PLC:




                    << Press ENTER Key to Start Store Function >>
ID:            RUN/OUT EN    23ms SCAN  ONLINE  L4 ACC: WRITE LOGIC    LOGIC EQUAL
C:\LM90\LESSON                        PRG: LESSON
REPLACE
```

The screen shows the currently selected program folder, which cannot be changed.

Three types of data can be stored from the PLC to the programmer: program logic, configuration data, or reference tables. When this screen first appears, only the program logic is set to **Y** (Yes). To store all of the data, change the selection for reference tables and configuration to **Y** (Yes). To store only part of the data, select **N** (No) for any of the following:

Field	Description
Program Logic	The ladder logic program.
Configuration	The current configuration.
Reference Tables	The reference tables for the program.

Note

Annotation files (nicknames, reference descriptions, and comment text) remain in the folder and are not stored to the PLC.

A program containing subroutine blocks and/or subroutine CALL instructions cannot be stored to a Pre-Release 3 PLC.

Use the cursor keys to select items, and type in new selections as appropriate. To restore the original selections while editing this screen, press **ALT-A**.

The information to be transferred must fit within the configured boundaries of the PLC (for example, its register memory size).

To begin storing the data, press the **Enter** key. The program must be complete, and must not contain errors in syntax or any instructions which are not supported by the attached PLC. If there are errors, the store operation is not attempted.

After a successful transfer of data, the software displays the message "Store Complete". If a communication or disk error occurs during the store process (indicated by a message on the screen), the selected items are cleared from the current folder. Correct the error and repeat the store function.

To stop a program transfer in progress, press **ALT-A** if the PLC is in **STOP** mode. If the PLC is in **RUN** mode when the store begins, you cannot abort the transfer of program logic.

To return to the Program Utility Functions menu, press the **Escape** key.

Note

If you store a program that was written using a CPU hardware configuration other than model 351 to a 351 CPU, you will be prompted to convert the folder to one that uses the code written for 351 CPUs when you start the storing process. Refer to page 7-4 for other considerations when using previously created folders with a 351 CPU.

Run Mode Store Function

The **RUN MODE STORE** function enables you to transfer program logic and reference table data to a running PLC system. To do this, the Logicmaster 90-30/20/Micro software automatically switches the PLC into **STOP/IOSCAN** mode during the transfer and then switches it back to **RUN** mode after the transfer is complete. The logic sweep is stopped for approximately 1 to 10 seconds, depending on the size of the program and the baud rate. (A very large program combined with a very low baud rate could result in more than 10 seconds.)

Caution

With Release 3.50 or earlier of Logicmaster 90-30/20 software and a CPU Release 3.x or earlier, non-retentive outputs are maintained during the store operation, but they are not maintained after the store operation is complete, as when the system toggles from STOP/IOSCAN mode back to RUN mode. All non-retentive outputs are turned OFF, regardless of their prior state, during this STOP-TO-RUN transition.

Retentive outputs are not affected by this operation. They are maintained at all times during and after the RUN MODE STORE.

Beginning with Release 4.01 of Logicmaster 90-30/20 software and Release 4.x CPUs, a new PLC mode called "Pause" will prevent non-retentive outputs from being cleared. Pause mode will cause the CPU sweep to be suspended and all I/O states frozen until the entire program has been transferred. The PLC will then resume operation where it paused, without the **STOP-TO-RUN** transition that clears non-retentive outputs in CPUs prior to Release 4.x.

Verifying a Program with the PLC

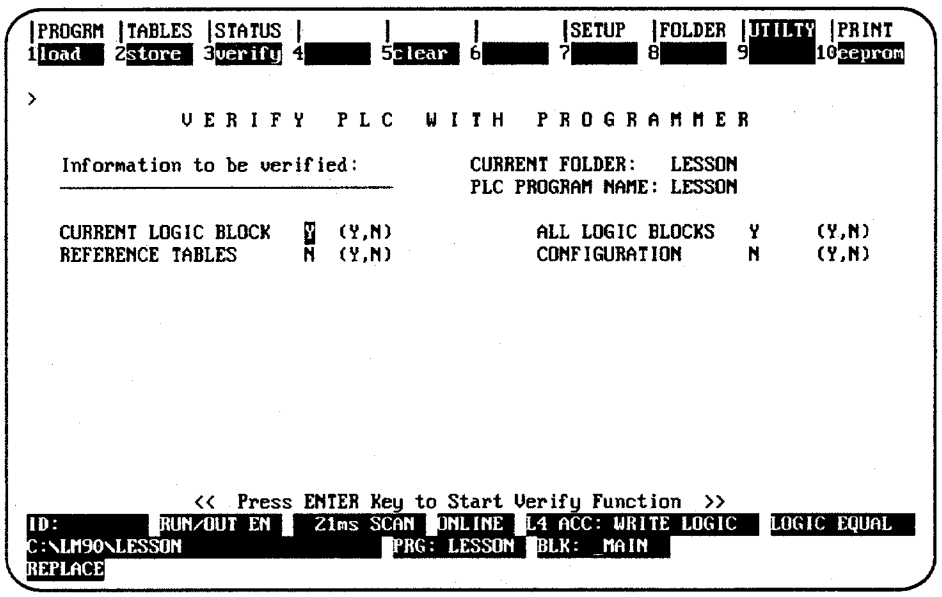
Use the verify function to compare program logic, configuration data, and/or reference tables in the programmer with the program logic, configuration data, and/or reference tables in the PLC. This function may be password protected. If so, you must know the password in order to use the function.

Note

You cannot verify a view locked block with an unlocked block. For more information on viewing locked blocks, refer to chapter 3, section 8, "Subroutine Blocks."

In the configuration software, only the configuration may be verified. No operations on program logic or tables may be performed.

To use the verify function, the programmer must be in **ONLINE** or **MONITOR** mode. Press **Verify** (**F3**) from the Program Utility Functions menu or from another Program Utilities screen.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1load   2store  3verify 4       5clear  6       7       8       9       10eeprom
>
            V E R I F Y   P L C   W I T H   P R O G R A M M E R

  Information to be verified:              CURRENT FOLDER:   LESSON
  ---------------------------              PLC PROGRAM NAME: LESSON

  CURRENT LOGIC BLOCK   Y  (Y,N)              ALL LOGIC BLOCKS   Y    (Y,N)
  REFERENCE TABLES      N  (Y,N)              CONFIGURATION      N    (Y,N)





             << Press ENTER Key to Start Verify Function >>
ID:          RUN/OUT EN    21ms SCAN   ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
C:\LM90\LESSON                        PRG: LESSON  BLK: _MAIN
REPLACE
```

The screen shows the currently selected program folder, which cannot be changed.

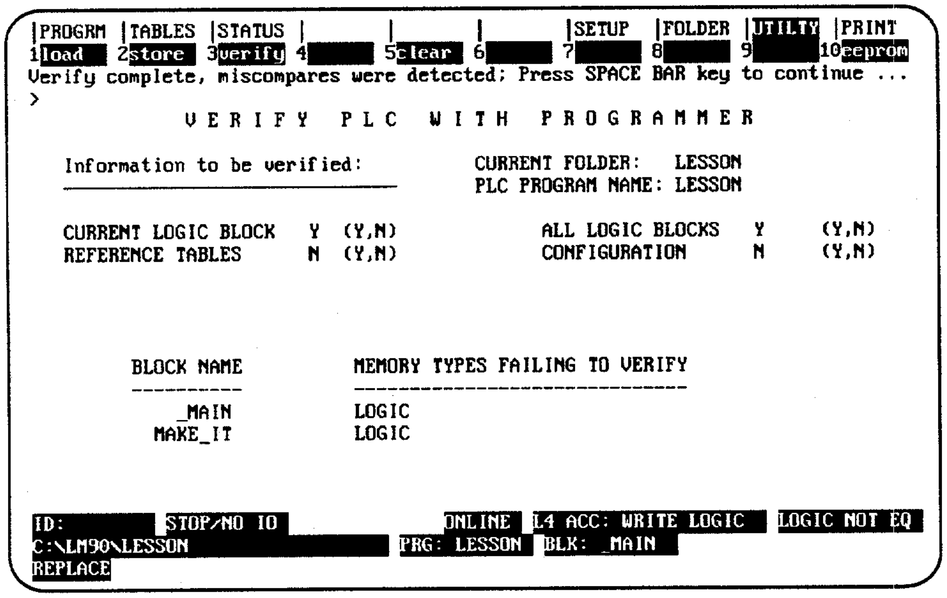
Four types of data can be compared: program logic, configuration data, reference tables, or all logic blocks. When this screen first appears, the selections for program logic and all logic blocks are set to **Y** (Yes). To compare all of the data, you must change the selection for configuration data and reference tables to **Y** (Yes). To compare only part of the data, select **N** (No) for any of the following:

Field	Description
Current Logic Block	The block currently selected in the program display/edit function.
All Logic Blocks	The main program block and all of the subroutine blocks (all blocks in the program).
Reference Tables	The reference tables for the program.
Configuration	The current configuration.

Press the **Enter** key to begin the verify function. While the verify is occurring, you can press **ALT-A** to abort.

If the information being compared is the same, the screen prompts: "Verify complete; no miscompares detected" and the program logic equality state is set to EQUAL. If the program is not successfully verified, the logic equality state is set to NOT EQUAL.

If miscompares are found, they are listed on the screen. For example:


```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1load   2store  3verify 4       5clear  6       7       8       9       10eeprom
Verify complete, miscompares were detected; Press SPACE BAR key to continue ...
>
              V E R I F Y   P L C   W I T H   P R O G R A M M E R

   Information to be verified:             CURRENT FOLDER:   LESSON
   ___________________________             PLC PROGRAM NAME: LESSON

   CURRENT LOGIC BLOCK   Y  (Y,N)              ALL LOGIC BLOCKS   Y     (Y,N)
   REFERENCE TABLES      N  (Y,N)              CONFIGURATION      N     (Y,N)



         BLOCK NAME          MEMORY TYPES FAILING TO VERIFY
         ----------          ------------------------------
              _MAIN          LOGIC
            MAKE_IT          LOGIC


ID:          STOP/NO IO              ONLINE  L4 ACC: WRITE LOGIC   LOGIC NOT EQ
C:\LM90\LESSON                      PRG: LESSON  BLK: _MAIN
REPLACE
```

Each miscompare displays the name of the logic block and the memory type where the miscompare occurs. Pressing the keyboard space bar will restore the screen to its original form or bring up the next page of miscompare information.

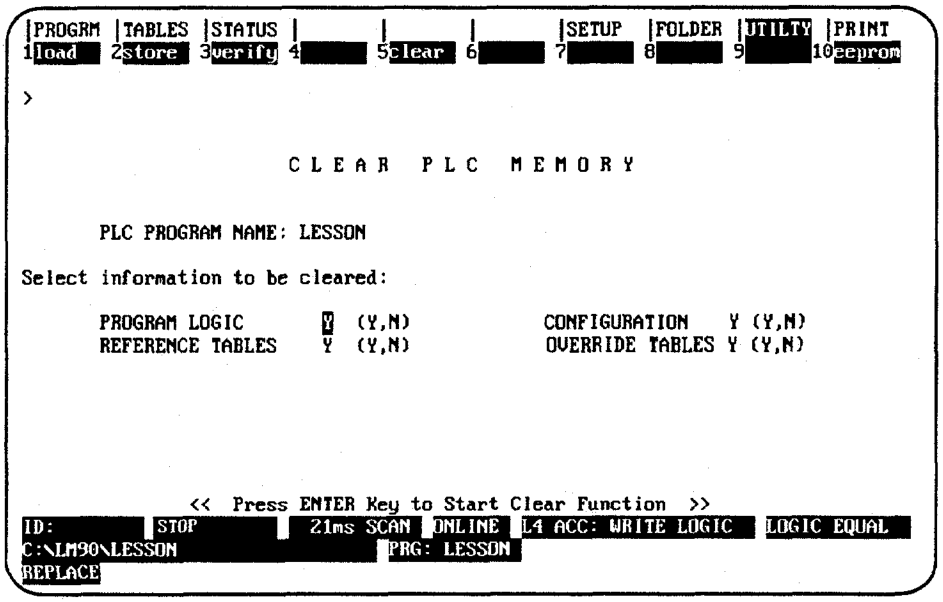
Clearing PLC Memory

The Clear PLC Memory function gives you the ability to selectively clear the program logic, configuration, reference tables, and/or reference override tables in the attached PLC.

Note

Only Model 331 and higher CPUs support reference overrides; therefore, this is nonapplicable to Models 211, 311, 313, 321, 323, and the Micro.

This function may be password protected in the PLC. If so, you must know the password to use the function. To use the clear function, the programmer must be in **ONLINE** mode. Press **Clear** (**F5**) from the Program Utility Functions menu or from another Program Utilities screen to reach the screen shown below:


|PROGRM |TABLES |STATUS | | | |SETUP |FOLDER |UTILTY |PRINT
1load 2store 3verify 4 5clear 6 7 8 9 10eeprom

```
>

                    C L E A R   P L C   M E M O R Y

       PLC PROGRAM NAME: LESSON

Select information to be cleared:

       PROGRAM LOGIC      Y  (Y,N)          CONFIGURATION   Y (Y,N)
       REFERENCE TABLES   Y  (Y,N)          OVERRIDE TABLES Y (Y,N)



              << Press ENTER Key to Start Clear Function >>
ID:          STOP          21ms SCAN ONLINE L4 ACC: WRITE LOGIC  LOGIC EQUAL
C:\LM90\LESSON                      PRG: LESSON
REPLACE
```

Four types of data can be cleared: program logic, configuration data, reference tables, and reference override tables. When this screen first appears, the selection for each type is set to **Y** (Yes). To clear all of the data, do not change any of these selections. When all options are set to **Y**, the %S fault bits and overrides are cleared. When only the reference tables option is set to **Y**, %S bits and overrides are ***not*** cleared. Select **N** (No) for those you do not want to clear.

Field	Description
Program Logic	The ladder logic program
Configuration	The current configuration
Reference Tables	The reference tables for the program
Override Tables	The reference override tables for the program

Press the **Enter** key to begin the clear function. ***Once the clear operation has begun, it cannot be aborted.*** When the clear operation is complete, the software displays the "Clear Complete" message.

Clear Reference Override Tables

The Clear PLC Memory function gives you the ability to selectively clear the program logic, configuration, and/or reference tables in the attached PLC. Beginning with Release 4.5, you can clear Reference Override Tables as well.

Special Considerations for Clearing Reference Overrides

The override tables in PLC memory are as follows: Input (%I), Output (%Q), User Internal (%M), and Global (%G). When you select "OVERRIDE TABLES" on the CLEAR PLC MEMORY screen, all reference override tables will be cleared in PLC memory.

Note

Do not confuse this with clearing the reference tables. To clear reference tables, set the "REFERENCE TABLES" to **Y** (Yes) as discussed on the previous page.

Restrictions

To clear the reference override tables, the CPU must be in STOP mode and online with Logicmaster. If you are using passwords with associated privileges, you must have a privilege level of 3 or above to use this feature.

Read/Write/Verify EEPROM/Flash

The Series 90-30 PLC supports an EEPROM (Electronic Erasable PROM) for storing PLC logic, configuration data, register data, passwords, and the OEM key. This EEPROM is located on the Series 90-30 PLC.

Note

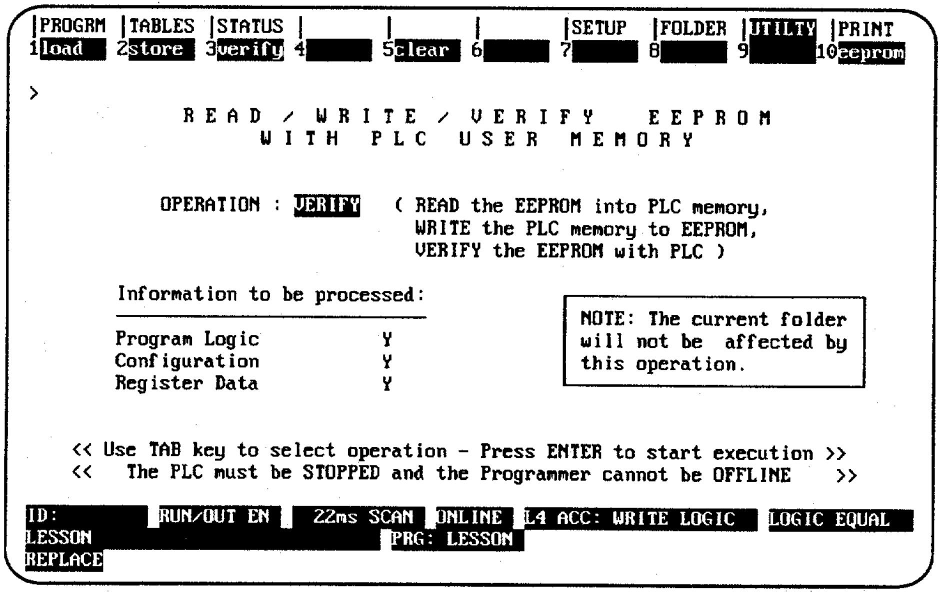
CPUs for the Micro as well as Model 340 and higher 90-30 CPUs have Flash memory instead of EEPROM; however, the Read/Write/Verify process is the same for Flash as for EEPROM. That is, all of the instructions listed below apply to all models having Flash memory, as well as all models having EEPROM.

The read/write/verify EEPROM function of Logicmaster 90-30/20/Micro software enables you to selectively read the EEPROM contents into PLC memory, write the entire contents of PLC memory to the EEPROM, and selectively verify the EEPROM contents with the PLC memory.

Note

The read/write EEPROM operations require the programmer to be online and the PLC to be stopped. The verify EEPROM operation can be performed while the programmer is in either **ONLINE** or **MONITOR** mode. In addition, these operations cannot be performed if the PLC is in OEM protection mode.

To use this function, press the **EEPROM** (**F10**) softkey from the Program Utility Functions menu or from another Program Utilities screen.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1load   2store  3verify 4       5clear  6       7       8       9       10eeprom
>
               R E A D  /  W R I T E  /  V E R I F Y    E E P R O M
                    W I T H   P L C   U S E R   M E M O R Y

             OPERATION : VERIFY   ( READ the EEPROM into PLC memory,
                                    WRITE the PLC memory to EEPROM,
                                    VERIFY the EEPROM with PLC )

           Information to be processed:
           ____________________________         NOTE: The current folder
           Program Logic           Y            will not be  affected by
           Configuration           Y            this operation.
           Register Data           Y

        << Use TAB key to select operation - Press ENTER to start execution >>
        <<   The PLC must be STOPPED and the Programmer cannot be OFFLINE    >>
ID:          RUN/OUT EN    22ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   LOGIC EQUAL
LESSON                              PRG: LESSON
REPLACE
```

When the Read/Write/Verify EEPROM screen is first displayed, the *Operation* field defaults to VERIFY. Use the Tab key to toggle through the valid operations, or enter the desired operation directly into the field. Then, select the information to be processed or just press the **Enter** key to start execution.

Operation	Description
Read EEPROM	Selectively read the EEPROM contents into PLC memory. Once the read operation is selected, you may choose the information to be read or use the default options provided: Program Logic: The ladder logic program. Configuration: The current configuration Register Data: %R data. When this screen first appears, all three types of data default to **Y** (Yes). Enter **N** (No) for those types of data you do not wish to read. After all the options are selected, press the **Enter** key (with the PLC connected and stopped). The software will prompt you to confirm this operation. *Once the read operation has begun, it cannot be aborted.* When the read operation is complete, the software displays the message "Read from EE-PROM completed" to indicate that the read was successful.
Write EEPROM	Write the entire contents of PLC memory into EEPROM. Once the write operation is selected, all of the selective option fields default to **Y** (Yes); these fields can be changed for the write operation. Press the **Enter** key (with the PLC connected and stopped) to begin the write operation. *Once the write operation has begun, it cannot be aborted.* When the write operation is complete, the software displays the message "Write to EEPROM completed" to indicate that the write was successful.
Verify EEPROM	Selectively verify the EEPROM contents with PLC memory. Once the verify operation is selected, you may choose the information to be read or use the default options provided: • Program Logic: The ladder logic program. • Configuration: The current configuration • Register Data: %R data. When this screen first appears, all three types of data default to **Y** (Yes). Enter **N** (No) for those types of data you do not wish to read. After all the options are selected, press the **Enter** key (with the PLC connected and stopped). If no miscompares are found, the message "Verify EEPROM complete; no miscompares found" is displayed on the message line. If miscompares are found, a corresponding error message is displayed to indicate which option(s) miscompared.

Chapter 9 Print Functions

The print function is used to:

- Determine a printing device or file destination for Logicmaster 90-30/20/Micro screen print text.
- Enter printer parameters.
- Print copies of programs, configurations, and reference tables to files or a printer.

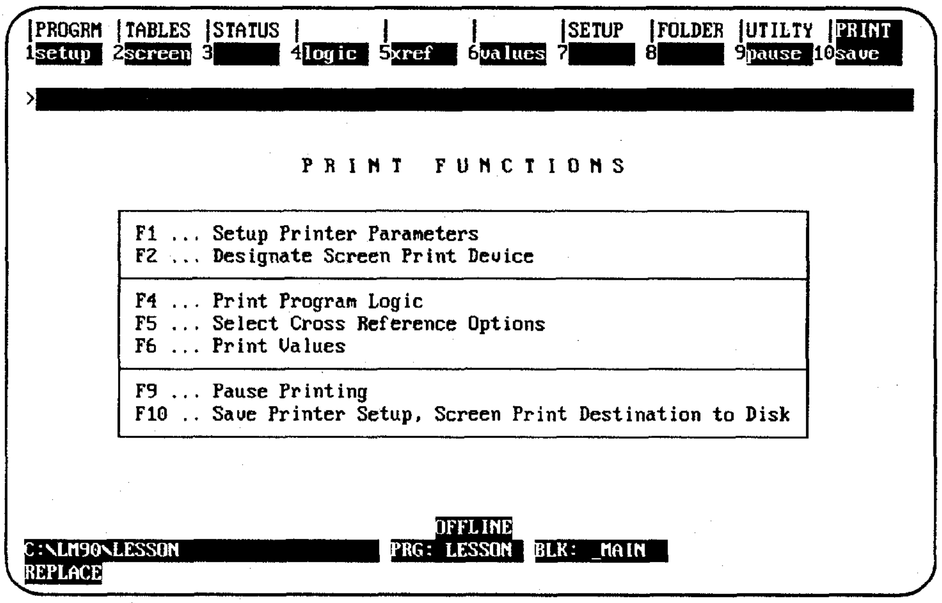
Note

Print may be used to print the values in the program folder only. To print the values from the PLC, the reference values must first be loaded to the folder by using the load utility function (see chapter 8).

Chapter 9 contains the following sections:

Print Functions Menu

To display the Print Functions menu in the programming software, press **Print** (**F10**) from the main menu, or **Shift-F10** from any main menu function screen.



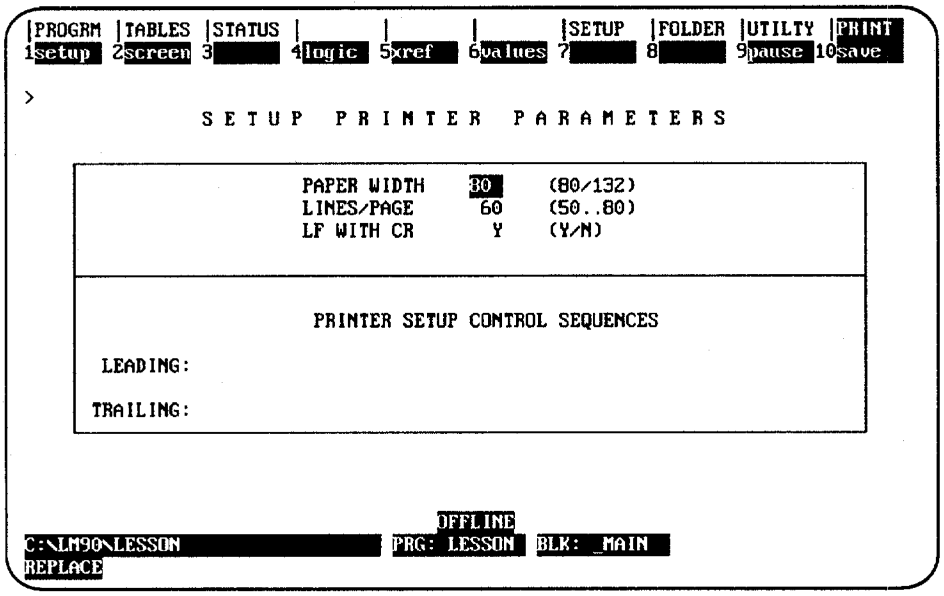
```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1setup  2screen 3       4logic  5xref   6values 7       8       9pause  10save
>

                         P R I N T   F U N C T I O N S

          F1 ... Setup Printer Parameters
          F2 ... Designate Screen Print Device

          F4 ... Print Program Logic
          F5 ... Select Cross Reference Options
          F6 ... Print Values

          F9 ... Pause Printing
          F10 .. Save Printer Setup, Screen Print Destination to Disk

                                          OFFLINE
C:\LM90\LESSON                            PRG: LESSON  BLK: _MAIN
REPLACE
```

Caution

Do not use the same file extensions for printing as are used for the program folder files. Logicmaster 90-30/20/Micro software does not prevent the printing of a listing to a file with the same name as one of the program folder files. If this is done, part of the folder may be lost. Refer to appendix G for a list of file extensions used by the software.

Section 1: Printer Parameters

The printer setup parameters used by the programmer must match those of your printer. Default values are supplied. These can be changed and stored in a file for continued use. To change your printer setup or check its contents, press **Setup** (**F1**) from the Print Functions menu or from another Print Functions screen.


|PROGRM |TABLES |STATUS | | | |SETUP |FOLDER |UTILTY |PRINT
1setup 2screen 3 4logic 5xref 6values 7 8 9pause 10save

```
>
                SETUP PRINTER PARAMETERS

                     PAPER WIDTH    80   (80/132)
                     LINES/PAGE     60   (50..80)
                     LF WITH CR      Y   (Y/N)

                PRINTER SETUP CONTROL SEQUENCES
  LEADING:

  TRAILING:

                                   OFFLINE
C:\LM90\LESSON                     PRG: LESSON  BLK: _MAIN
REPLACE
```

The setup screen shows the value and possible selections for each parameter.

Changing Setup Printer Parameters

To change entries on the Printer Parameters screen:

1. Move the cursor to the item you want to change.
2. Type (or use the **Tab** key) to enter new values as needed.
3. When you are finished, decide whether your entries or the default values shown above should be used when the system is started.

   A. To save the printer parameters only until the next power-up (and then return to the default parameters), do not press **Save** (**F10**).

   B. To save the printer parameters, press **Save** (**F10**). This causes the printer parameters to be written to the PRINT.SET file. This file is written to the home directory. When the programmer is powered up, these entries will replace the standard default values.

Printer Parameters

Refer to the following descriptions when changing the entries on the Printer Parameters screen.

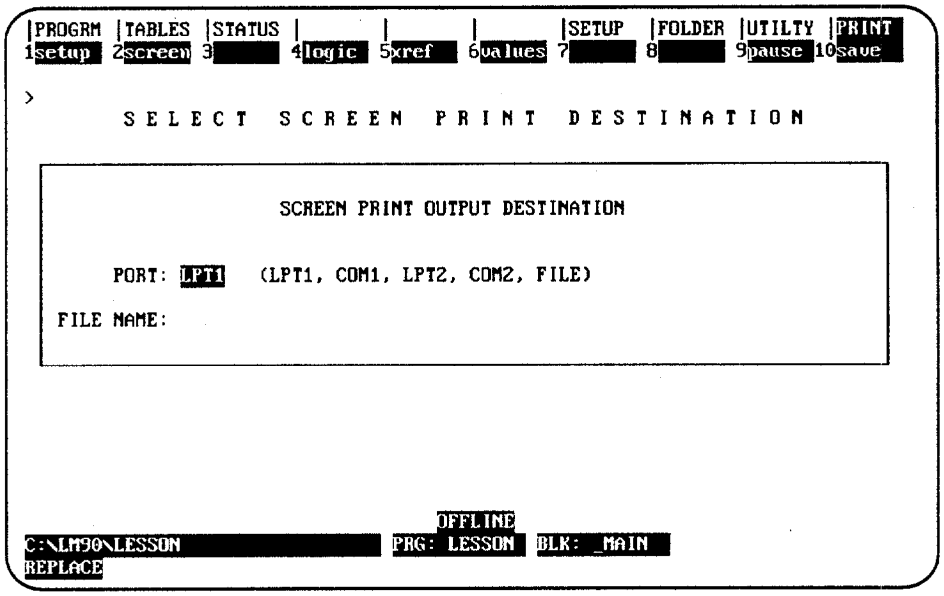
Table 9-1. Printer Parameters

Parameter	Description
Paper Width	The number of characters printed on a line. If the printer is set up for standard 8-1/2 inch wide paper, select 80 characters for the paper width. If the printer uses 11-inch (or wider) paper, you may select either 80 or 132 characters. In 80-character mode, rung numbers and cross references are printed above the rungs. In 132-character mode, the format of the ladder logic is the same. However, rung numbers and cross references are printed to the right of the rungs.
Number of Lines per Page	The number of lines (from 50 to 120) that can be printed on a page (default=60). If an associated group of lines will not fit on one page, the system will command the printer to advance to the next page after the number of lines specified by this entry. Note: Some printers automatically insert a formfeed after printing a certain number of lines (typically, 66). If the printer has this feature, specify a shorter page length to prevent an automatic page eject.
Line Feed with a Carriage Return	The line feed character advances the paper to the next line for printing. This item determines whether the system automatically inserts a line feed (<CR><LF>) each time the printer head should return to the left page margin. Note: Some printers can be set up to automatically advance to the next line by entering only <CR>. Refer to the instructions for your printer to determine how it works. If you do not want the programmer to insert a line feed character after each carriage return character, enter **N** (No). To have the programmer insert the <LF> character after each <CR> character, enter **Y** (Yes).
Printer Setup Sequences	If the printer uses leading or trailing control characters, enter the characters here. If you move the cursor to either of these entries, a field will appear where you can enter the appropriate characters. Printer control sequences are issued immediately before and after a listing. This feature can be used to put the printer into a particular mode (for example, compressed output), returning to the original mode after the listing is finished. From 0 to 60 characters can be specified. To enter non-printing characters, use one backslash followed by the three-digit decimal equivalent of the ASCII representative. For example, to identify the Escape character <ESC>, you would enter `\027`. A leading zero is required. If you need to enter the backslash character itself, enter two backslashes: `\\`.

Section 2: Selecting a Screen Print Device

The print screen function can be used to print a copy of any screen in the programmer. To execute a screen print, go to the screen you wish to print and press **ALT-P**. The printing of the screen is sent to the selected device or file. To abort the printing before it completes, press any key.

Screen prints can be sent to an output port or directed to a file. To specify a screen print destination, press **Screen** (**F2**) from the Print Functions menu or from another Print Functions screen.


```
|PROGRM |TABLES |STATUS |        |        |        |SETUP   |FOLDER  |UTILTY  |PRINT
1setup  2screen 3        4logic  5xref   6values 7        8        9pause  10save

>
        S E L E C T   S C R E E N   P R I N T   D E S T I N A T I O N

                     SCREEN PRINT OUTPUT DESTINATION

        PORT: LPT1   (LPT1, COM1, LPT2, COM2, FILE)

    FILE NAME:



                                               OFFLINE
C:\LM90\LESSON                             PRG: LESSON  BLK: _MAIN
REPLACE
```

Sending Screen Prints to a Printer

If screen prints should be sent to a printer, the selection for the port should match the current printer port. If the port is a serial port (COM1 or COM2), be sure to set the baud rate using the program setup function. To use COM2, the PC must have two serial ports physically present. (This is an MS-DOS restriction.)

To save the port designation, press **Save** (**F10**). When you exit the screen, the programmer will create a file named SCRPRINT.SET, located in the home directory. This file is read by the programmer at startup. To use the port designation temporarily, exit the screen without pressing **Save**. The port selected is used until the programmer is powered down or until changed on this screen.

Sending Screens to a File

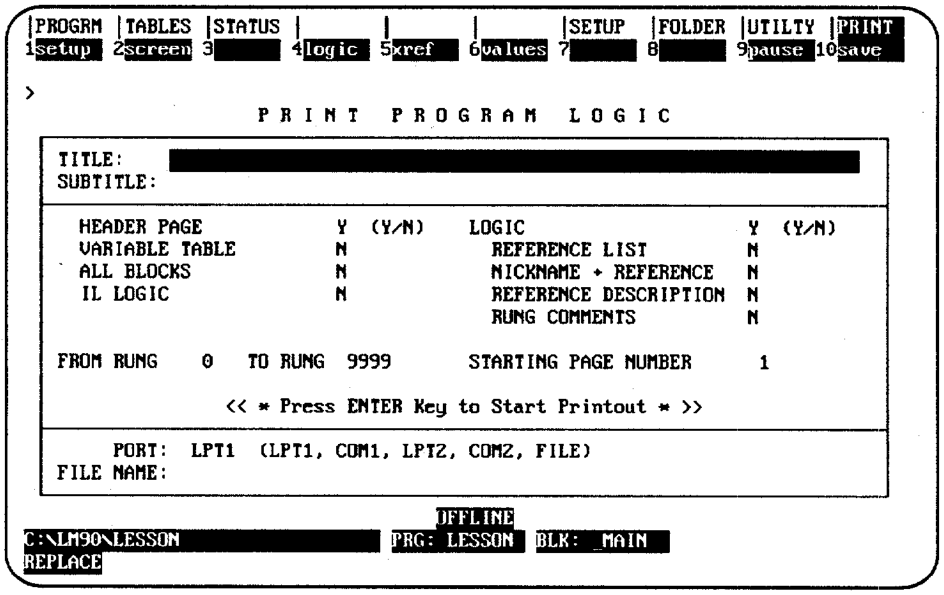
If screen prints should be sent to a file, then the port entry should be "file". You can enter any valid file name for the screen print file. This file is used to store all future screen prints, which are added to the end of the file.

When specifying the screen print device destination to disk, use the full path name. If only a file name is specified, the screen print file is written to the folder that was active when the screen print device was specified; this may or may not be the current folder. Refer to appendix G for a listing of reserved file extensions.

If printing is paused while printing to a floppy disk, the disk swapped for a second disk, and then printing is resumed, neither disk will end up with a usable listing file. In addition, the second disk must be formatted before further use.

Section 3: Print Program

The print program function is used to print program logic, cross references, variables, and program annotation. To use the print program function, press **Logic** (**F4**) from the Print Functions menu or from another Print Functions screen.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1setup  2screen 3       4logic  5xref   6values 7       8       9pause  10save
>
                    P R I N T   P R O G R A M   L O G I C

 TITLE:
 SUBTITLE:

   HEADER PAGE            Y  (Y/N)   LOGIC                     Y  (Y/N)
   VARIABLE TABLE         N            REFERENCE LIST          N
   ALL BLOCKS             N            NICKNAME + REFERENCE    N
   IL LOGIC               N            REFERENCE DESCRIPTION   N
                                       RUNG COMMENTS           N

 FROM RUNG    0   TO RUNG  9999      STARTING PAGE NUMBER      1

               << * Press ENTER Key to Start Printout * >>

      PORT:  LPT1  (LPT1, COM1, LPT2, COM2, FILE)
 FILE NAME:
                                        OFFLINE
C:\LM90\LESSON                         PRG: LESSON  BLK: _MAIN
REPLACE
```

Specify the content for the printout on this screen as instructed on the following pages. If you do not change a selection, its default is used. If the printout should also include program cross references (which show other program uses of selected references), press **F5** to select the cross references to be included.

To save the content of this screen to a file called PSCREENS.SET, press **Save** (**F10**). This file is located in the directory of the active program folder and is read when the print function is entered. If you do not save the content of this screen, the default values will be used the next time you print program logic, cross references, variables, or program annotation.

After completing the selections, press the **Enter** key to start printing. Printing will continue unless paused or aborted. When the printing is complete, the software displays the message "Listing complete".

To temporarily suspend printing after it begins, press **Pause** (**F9**). To restart printing, press **F9** again.

An attempt to print to a printer that is out of paper results in the message, "Printing device is offline; press pause to resume," being displayed.

To abort printing while in progress, press **ALT-A** and respond to the prompt. Printing will also abort if you press **CTRL-Break**, which will exit the programming software. When a listing is being printed to a file and the listing is aborted, the file containing the listing up to that point is closed and is not deleted. It contains valid data up to the point where it was stopped.

Print Program Parameters

Refer to the following definitions when changing the entries on the Print Program page. Examples are provided in section 6 of this chapter.

Table 9-2. Print Program Parameters

Parameter	Description
Title	An optional title of up to 62 characters centered at the top of each page. The title may change if specified in a rung comment.
Subtitle	An optional line of up to 62 characters printed below the title on each page. The subtitle may change if specified in a rung comment.
Header Page	The header page for a program shows the program name, CPU reference sizes, highest references used in the program, and the size of the program,in bytes. To print header pages, enter **Y** (Yes).
Variable Table	To print tables of variables and identifiers, enter Y (Yes).
All Blocks	To print all blocks (except those that have been view locked) in the selected program, enter Y (Yes). During listing or printing, the name of the block being printed appears on the status line at the bottom of the screen. Logic for blocks which have been view locked cannot be printed, even if **Y** (Yes) is selected for ***All Blocks***. A list of the view locked blocks that could not be printed will be included at the end of the printout. For more information on locked blocks, refer to chapter 3, section 8, "Subroutine Blocks." If ***All Blocks*** is set to **N** (No), only the current block is printed. The name of the block is displayed on the status line. Whether one block or all blocks are printed, the printout will include the IL Logic and Logic items specified below.
IL Logic	To print Instruction List logic, enter **Y** (Yes).
Logic	To print ladder diagram logic, enter **Y** (Yes). Then, specify the content of the logic printout: Reference List: Enter **Y** (Yes) to print a table at the bottom of a page listing all the program references on that page. For each reference, the table lists the machine address, nickname, and reference description. Nickname and Reference: Enter **Y** (Yes) to print both machine addresses and their nicknames with the ladder logic. If **N** (No) is entered, only nicknames are printed in the logic. Machine addresses are printed for any references that do not have nicknames assigned. Reference Description: Enter **Y** (Yes) to print reference descriptions above the reference addresses at the next print logic request. When the print reference description feature is selected, the first 28 characters of the reference description for each reference address in the logic are printed above the reference address. The reference description is divided into four lines of seven characters each. If the nickname and reference option is also selected, the reference description is printed above the nickname, for a total of six lines above the instruction. If the nickname and reference option is not selected, the reference description is printed directly above the reference address or nickname, for a total of five lines above the instruction. Refer to chapter 3, section 7, "Changing the Display Mode," and chapter 6, "Programmer Setup," for more information on selecting the view modes. Rung Comments: Enter **N** (No) to print comments in instruction form only. If **Y** (Yes) is entered, rung explanation text is printed in place of the comment instruction. Additional text for rung comments may be created as separate files, as described in chapter 3, section 4, "Program Annotation." If this has been done, be sure the files are present on the specified drive when printing begins.

Table 9-2. Print Program Parameters (cont'd)

Parameter	Description
From/To Rung	To print a range of logic rungs within the program, enter the first and last rung number of the group to be printed. If the *All Blocks* option is set to **Y** (Yes), these fields will be ignored.
Starting Page Number	To begin numbering the printout with a page number other than 1, enter the number here.
Port	Specify the destination of the listing.
File Name	To send the printout to a file instead of to a printer, enter a name for the file here. A printout file cannot be written to a locked folder. If a file already exists with the selected name, the new file will overwrite it. Refer to appendix G for a listing of the extensions used by Logicmaster 90-30/20/Micro software. You do not want to overwrite one of these files with print text.

Shortcuts for Printing Program Logic

The following suggestions can help you shorten a program logic printout:

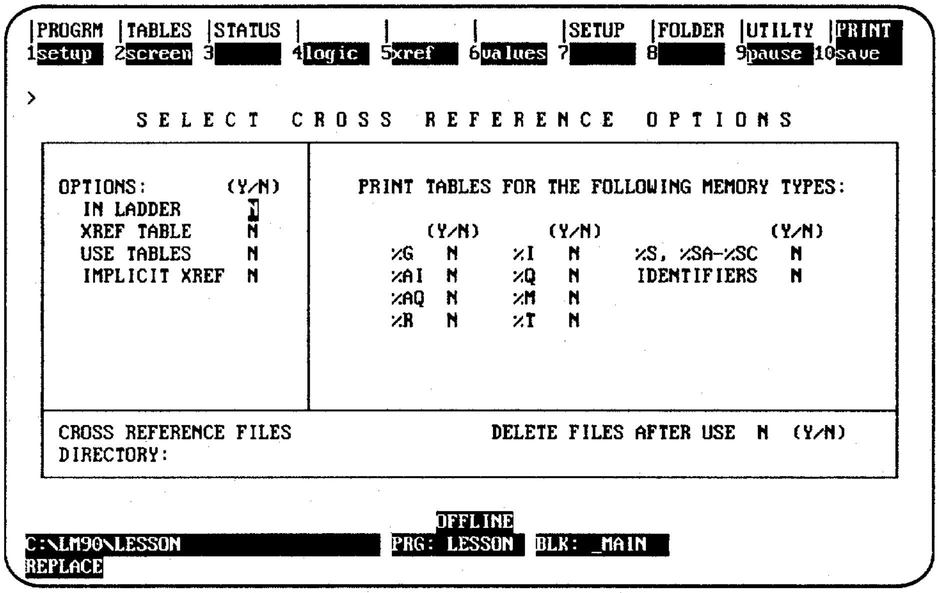
1. Do not select both the reference list and reference description options.
2. If the program contains many reference descriptions, use the reference list option instead of the reference description option.
3. If the program contains few reference descriptions, use the reference description option instead of the reference list option.
4. Use the 132-column page width instead of the 80-column page width. This is very helpful when printing in-ladder cross references with logic.

Cross References

Cross references in a printout show the use of references in the program. Cross references can be included as part of the ladder logic text and/or listed as separate tables.

If the *All Blocks* parameter on the Print Program Logic screen is set to **Y** (Yes), the cross reference tables for each block are printed following the logic for that block.

To include cross references, press **F5** from the Print Program screen or from another Print Functions screen.



```
|PROGRM |TABLES |STATUS |        |        |        |SETUP  |FOLDER |UTILTY |PRINT
1setup  2screen 3       4logic  5xref   6values 7       8       9pause  10save

>
        S E L E C T   C R O S S   R E F E R E N C E   O P T I O N S

 OPTIONS:        (Y/N)     PRINT TABLES FOR THE FOLLOWING MEMORY TYPES:
   IN LADDER       N
   XREF TABLE      N            (Y/N)      (Y/N)                  (Y/N)
   USE TABLES      N        %G    N     %I   N     %S, %SA-%SC     N
   IMPLICIT XREF   N        %AI   N     %Q   N     IDENTIFIERS     N
                            %AQ   N     %M   N
                            %R    N     %T   N

 CROSS REFERENCE FILES                    DELETE FILES AFTER USE  N  (Y/N)
 DIRECTORY:

                                        OFFLINE
C:\LM90\LESSON                         PRG: LESSON  BLK: _MAIN
REPLACE
```

Specify the content and format of the cross reference printing on this screen. If you do not change a selection, its default is used.

To save the content of this screen to a file called PSCREENS.SET, press **Save** (**F10**). This file is located in the directory of the active program folder and is read when the print function is entered. If you do not save the content of this screen, the default values will be used the next time you print program logic, cross references, variables, or program annotation.

Press **Logic** (**F4**) from this screen to return to the Print Program Logic screen.

The following key can be used for interpreting cross reference instruction symbols:

Symbol	Description
###	Explicit reference.
(###)	Implicit reference.
FBIO	Function block direct reference.
JMP	Jump control.
LABEL	Label.
MCR	Master control relay.
EMCR	End master control relay.
CALL	Block CALL instruction.

Refer to the following definitions when changing the entries on the Cross Reference screen.

Table 9-3. Cross Reference Parameters

Parameter	Description
In Ladder	To include cross references as part of the ladder logic, enter **Y** (Yes). Cross references for each reference address used on the coils in a rung are printed with that rung. In addition, each time the reference address is used on a contact, the most recent rung on which the reference address was used on a coil (referred to as the controlling rung) is listed under the contact.
Cross Reference Table	Specify whether to print separate tables of cross references for each block and/or a single set of cross reference tables for the program. The printout contains a table for each reference type which has at least one address reference used anywhere in a program. Within each table, cross reference information is printed in ascending order of reference address offset. For each reference address offset, the cross reference data is grouped by block and, then within each block, by the instructions on which the reference is used. For each instruction, rung numbers are printed in ascending order. For references used more than once on a given rung in the same instruction, the rung number is listed only once. Per Block: If **Y** (Yes) is selected for *Per Block,* sets of unique cross reference tables for each block are printed. A per block cross reference table contains only the cross reference information for references used within the scope of a single block. The default for this option is **N** (No). Global: If **Y** (Yes) is selected for *Global,* one set of cross reference tables for the program is printed. A global cross reference table for a reference type will contain all the cross reference information for references of that type used anywhere within the scope of the program. The information for a reference address will consist of each block, rung number, and instruction on which the address was used. The information will also include the unique nickname and reference description assigned to the reference address within each block. The default for this option is **N** (No). The nickname and reference description defined for a reference address within the scope of a block is printed when the global cross reference data for that reference address and that block is printed. Implicit cross reference data may be included or excluded from either type of cross reference table listing by entering **Y** or **N** in the *Implicit Xref* field. If included, the implicit usage of a reference address by an instruction is printed with parentheses around the rung number of the rung containing that instruction. An identifier name global cross reference table will be printed when the *Xref Tables Global* option is selected. Since the only identifier table names known and used globally in a program are the block names, the global identifier name cross reference table will consist only of the usage information for block names. For each block name, the global cross reference identifier table will list the blocks in which it is called, and the rung numbers within each block of the rungs containing the CALL instruction. Global cross reference tables will be printed after all other logic listing text that you selected has been printed and before the global use tables.

Table 9-3. Cross Reference Parameters (cont'd)

Parameter	Description
Use Tables	Specify whether to print separate reference use tables for each block and/or a single set of reference use tables for the program. The symbol - means a reference is not used. The symbol * means used explicitly on an instruction operand. The symbol + means used implicitly by an instruction. When both * and + apply, the symbol # is used. (Refer to the entry for implicit cross references in this table for more information.) Per Block: If Y (Yes) is selected for *Per Block*, sets of unique reference use tables for each block are printed. A per block reference use table contains only the reference use information for references used within the scope of a single block. The default for this option is N (No). Global: If Y (Yes) is selected for *Global*, one set of reference use tables for the program is printed. A global reference use table for a reference type will contain reference use information for the entire program. The default for this option is N (No). Implicit reference usage data may be included or excluded from either type of reference use table listing by entering Y or N in the ***Implicit Xref*** field.
Implicit Cross Reference	Implicit references are references that are used, but which do not appear directly in the program. For example, if the MOVE_INT instruction has an input %AI001, an output %AQ001, and a length of 5, %AI001 and %AQ001 are explicitly used. Due to the length, there are really 5 words of input/output. The next 4 words are implicitly used. To include both implicit and explicit references in cross reference tables, enter Y (Yes). If N (No) is entered, only explicit references are printed.
Memory Types	Enter Y to select the reference types for which cross reference tables and/or reference use tables will be printed. The tables will begin at reference address offset 1 and end with the highest used reference address offset of each reference address type in the program or block. Enter N for those references whose cross reference tables and/or reference use tables will not be printed. To include cross references of block names, and MCR, ENDMCR, JUMP, and LABEL names in a cross reference table printout, enter Y (Yes) in the ***Identifiers*** field.

Table 9-3. Cross Reference Parameters (cont'd)

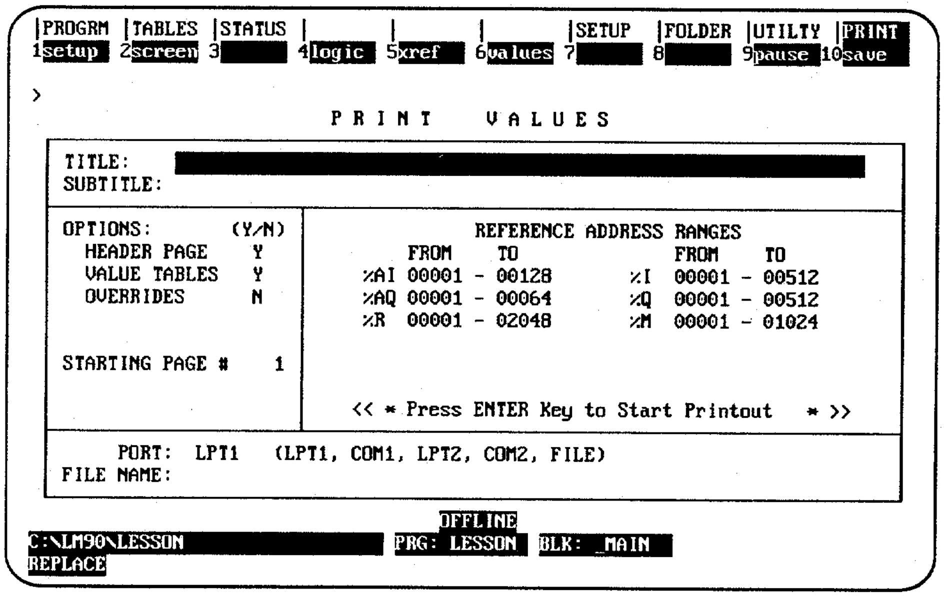
Parameter	Description
Directory	Specify a directory to store cross reference data files. The specified location for the .XRF, PRINT.XOV, and GLOBAL.PTX cross reference data files must be either a hard disk, an expanded memory device, or a RAM Disk. The cross reference files cannot be sent to a floppy disk or a locked folder. In either case, an error message is displayed. If no entry is made here, the files are stored in the current folder. Refer to appendix G for extensions used by Logicmaster 90-30/20/Micro software. Do not overwrite one of the Logicmaster 90 files with print text. Note: Cross reference data files are not automatically deleted when exiting the print function. The .XRF and PRINT.XOV files remain in the designated directory unless you enter **Y** (Yes) in the ***Delete Files After Use*** field. This prevents having to cross reference the same blocks each time a listing is started. When subsequent listings from the same folder are generated, only the cross reference data files for the blocks which have changed since the previous listing are deleted and recreated. The default selection for the ***Delete Files After Use*** field is **N** (No). In addition to the .XRF and PRINT.XOV files, a temporary file named GLOBAL.PTX is created. The file will exist during a listing that includes global cross reference text; it will be deleted when the global cross reference listing terminates successfully, terminates with an error, or aborts.
Delete Files After Use	To delete the cross reference files after the listing completes, enter **Y** (Yes). If you want the cross reference files to remain on disk, enter the default **N** (No). To delete cross reference files from a folder without printing a listing, use the selected clear folder function.

Section 4: Print Reference Tables

To print reference table values (excluding coil references) for the program, press **Values** (**F6**) from the Print Functions menu or from another Print Functions screen. (See the next section for printing coil references.)

Note

Print may be used to print the values in the program folder only. To print the values from the PLC, the reference values must first be loaded to the folder by using the load utility function (see chapter 8).


|PROGRM |TABLES |STATUS | | | |SETUP |FOLDER |UTILTY |PRINT
1setup 2screen 3 4logic 5xref 6values 7 8 9pause 10save

```
>
                        P R I N T    V A L U E S
 TITLE:
 SUBTITLE:

 OPTIONS:          (Y/N)          REFERENCE ADDRESS RANGES
   HEADER PAGE      Y          FROM     TO              FROM     TO
   VALUE TABLES     Y      %AI 00001 - 00128       %I  00001 - 00512
   OVERRIDES        N      %AQ 00001 - 00064       %Q  00001 - 00512
                           %R  00001 - 02048       %M  00001 - 01024

 STARTING PAGE #      1

                       << * Press ENTER Key to Start Printout   * >>

        PORT:  LPT1   (LPT1, COM1, LPT2, COM2, FILE)
 FILE NAME:
```

OFFLINE
C:\LM90\LESSON
PRG: LESSON
BLK: _MAIN
REPLACE

The listing can be directed to a printer or to a file.

These tables correspond to the tables that can be displayed and formatted using the reference tables function. For example:

```
                  *****   I N P U T   V A L U E   T A B L E   *****

REF.     |64       |56       |48       |40       |32       |24       |16       |8       |1
ADDRESS  +-------- +-------- +-------- +-------- +-------- +-------- +-------- +--------+

%I0064 01101001 00001000 00010101 01011011 00010000 11100100 11010100 01001110
%I0128 00001101 11101111 01100100 01111011 01000011 11001010 00010110 00011000
%I0192 01011111 11110010 00111000 11100011 00010001 10101011 00000000 00000000

%I0256 00000000 00000101 00010010 00011100 00000000 00101101 01011011 10011100
```

Specify the content of the printout on the Print Values screen.

Table 9-4. Value Table Parameters

Field	Description
Title	An optional title of up to 62 characters centered at the top of each page. The title may change if specified in a rung comment.
Subtitle	An optional line of up to 62 characters printed below the title on each page. The subtitle may change if specified in a rung comment.
Header Page	The header page for a program shows the program name, CPU reference sizes, highest references used in the program, and the size of the program, in bytes. To print header pages, enter **Y** (Yes).
Value Tables	To print value tables (input, output, etc.), enter **Y** (Yes). Value tables are printed in groups of three lines at a time.
Overrides	To print the override value tables associated with selected %I, %Q, and %M references, enter **Y** (Yes).
Starting Page	To begin numbering the printout with a page number other than 1, enter the number here.
Address Range	Specify the range of values to be printed. Enter values only for types of references to be printed. If any references should not be printed, enter zeros in both the *FROM* and *TO* fields. If the configured limits change (i.e., by changing folders, clearing a folder, or downloading from the PLC), the address ranges will be re-initialized.
Port	Specify the destination of the listing.
File Name	To send the printout to a file instead of to a printer, enter a name for the file here. A printout file cannot be written to a locked folder. If a file already exists with the selected name, the new file will overwrite it. Refer to appendix G for a listing of the extensions used by Logicmaster 90-30/20/Micro software. You do not want to overwrite one of these files with print text.

To save the content of this screen to a file called PSCREENS.SET, press **Save** (**F10**). This file is located in the directory of the active program folder and is read when the print function is entered. If you do not save the content of this screen, the default values will be used the next time you print program logic, cross references, variables, or program annotation.

After completing the selections, press the **Enter** key to start printing. Printing will continue unless paused or aborted. When the printing is complete, the software displays the message "Listing is complete".

To temporarily suspend printing after it begins, press **Pause** (**F9**). To restart printing, press **F9** again.

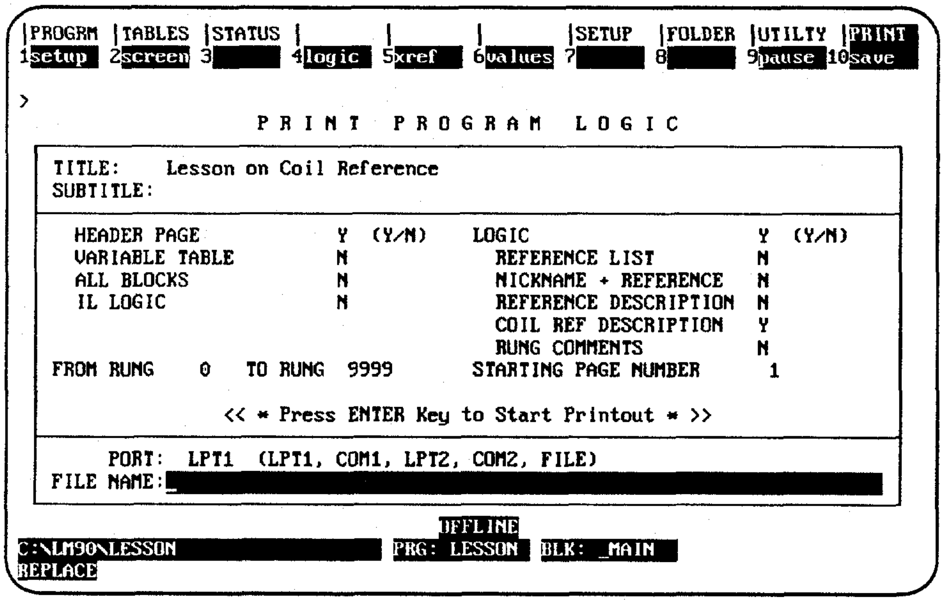
During printing, press **ALT-A** to abort the printout. Printing will also abort if you press **CTRL-Break** and confirm your action. This will exit the programming software.

Note

When a listing is being printed to a file and the listing is aborted, the file containing the listing up to that point is closed and is not deleted. It contains valid data up to the point where it was stopped.

Section 5: Print Coil References

You can print reference descriptions for coils through the new parameter "COIL REF DESCRIPTION" as shown below:



```
|PROGRM |TABLES |STATUS |        |        |        |SETUP   |FOLDER |UTILTY |PRINT
1setup  2screen 3        4logic   5xref    6values  7        8        9pause  10save
>
                       P R I N T   P R O G R A M   L O G I C

  TITLE:    Lesson on Coil Reference
  SUBTITLE:

    HEADER PAGE            Y  (Y/N)   LOGIC                     Y  (Y/N)
    VARIABLE TABLE         N            REFERENCE LIST          N
    ALL BLOCKS             N            NICKNAME + REFERENCE    N
    IL LOGIC               N            REFERENCE DESCRIPTION   N
                                        COIL REF DESCRIPTION    Y
                                        RUNG COMMENTS           N
  FROM RUNG    0   TO RUNG  9999      STARTING PAGE NUMBER      1

                  << * Press ENTER Key to Start Printout * >>

       PORT:  LPT1  (LPT1, COM1, LPT2, COM2, FILE)
  FILE NAME:_

                                      OFFLINE
C:\LM90\LESSON                       PRG: LESSON  BLK: _MAIN
REPLACE
```

The default for this print parameter is No. The "REFERENCE DESCRIPTION" option controls the printing of reference descriptions for all other references. The sample section from a 132-column print (shown below) resulted from the settings shown above. See Note below for information about the differences between 80- and 132-column print.

```
|AUTOPB   LISUP  %Q0038  %Q0012  %I0013  EMERGST %Q0006  %Q0075          %Q0005   << RUNG 4  STEP #0001 >>
+--] [-----] [-----] [-----] [--+--]/[-----]/[-----] [-----] [-------------( )--
|                               |
|%Q0005                         |
+--]/[--------------------------+
|
|EMSTOP  %I0100  CLAMPED %I0073                                             EMERGST  << RUNG 5  STEP #0011 >>
+--] [--+--] [--+--] [--+--] [---------------------------------------------( )-- (* Emergency Return         *)
|       |       |       |
|       |EMERGST|UNISUP |
|       +--]/[--+--]/[--+
|
|%Q0005  %Q0012  %I0021  %I0022                                             %M0001   << RUNG 6  STEP #0020 >>
+--]/[-----]/[-----] [-----] [---------------------------------------------( )--
```

Note

For 132-column listings, the coil reference descriptions print in a single line to the right of each coil's reference address as shown above. When set for an 80-column listing, the coil reference descriptions print in the standard LM90 reference description form of four lines above each reference address. See the table on the next page for additional considerations.

The following table is provided to explain the options you now have for printing references.

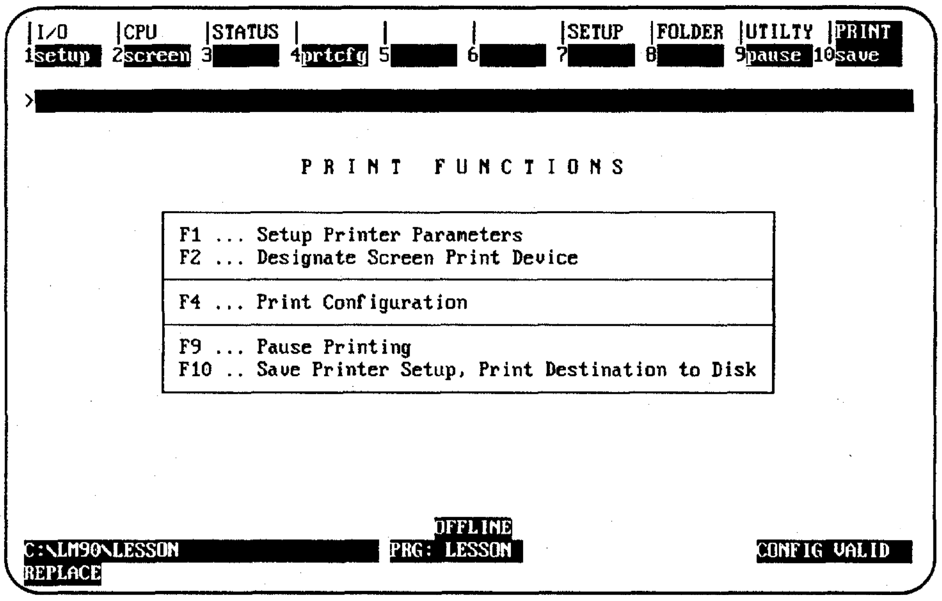
If you set the REFERENCE DESCRIPTION print option to:	And the COIL REF DESCRIPTION print option to:	The results are:
Y	N	Prints reference descriptions above contacts and instruction operands, but it does not print coil reference descriptions.
N	Y	Prints reference descriptions for coils only.
Y	Y	Prints reference descriptions for all references.

Section 6: Print Configuration

In the Logicmaster 90-30/20/Micro configuration software package, the print function enables you to obtain a listing of all I/O and CPU configuration data. The listing can be directed to a port for immediate printout, or to a file for later printing. The following functions are provided:

- View/change printer parameters.
- Select the screen print destination.
- Generate a rack hardware configuration listing.
- Generate a configured reference address listing (Ref View).
- Generate a CPU configuration listing.

To display the Print Functions menu in the configuration software, press **Print** (**F10**) from the main menu, or **Shift-F10** from any main menu function screen.

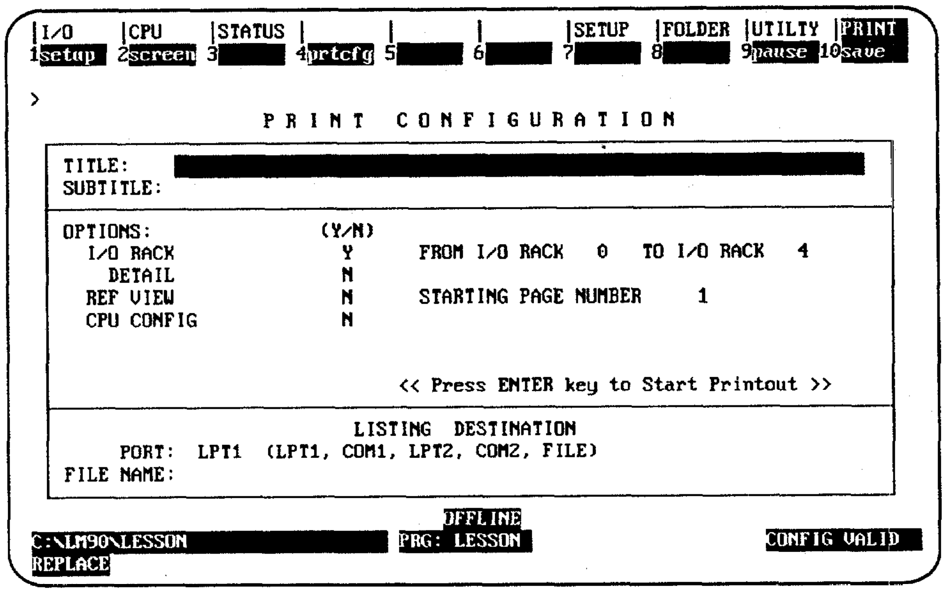


```
|I/O     |CPU     |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1setup   2screen  3       4prtcfg 5       6       7       8       9pause  10save

>

                         P R I N T   F U N C T I O N S

              F1 ... Setup Printer Parameters
              F2 ... Designate Screen Print Device

              F4 ... Print Configuration

              F9 ... Pause Printing
              F10 .. Save Printer Setup, Print Destination to Disk

                                          OFFLINE
C:\LM90\LESSON                           PRG: LESSON                CONFIG VALID
REPLACE
```

The Setup (F1) and Screen (F2) keys perform the same functions as their corresponding keys in the programming software. Refer to section 1 of this chapter for information on setting up the printer parameters, and to section 2 for information on selecting the screen print output destination.

Accessing the Print Configuration Screen

To use this function, press **Print Configuration** (**F4**) from the Print Configuration menu. The following screen will be displayed:



```
|I/O     |CPU     |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1setup   2screen  3       4prtcfg 5       6       7       8       9pause  10save

>
                      P R I N T   C O N F I G U R A T I O N

 TITLE:
 SUBTITLE:

 OPTIONS:              (Y/N)
   I/O RACK              Y      FROM I/O RACK   0   TO I/O RACK   4
     DETAIL              N
   REF VIEW              N      STARTING PAGE NUMBER      1
   CPU CONFIG            N

                                << Press ENTER key to Start Printout >>

                           LISTING  DESTINATION
      PORT:  LPT1  (LPT1, COM1, LPT2, COM2, FILE)
 FILE NAME:

                                       OFFLINE
C:\LM90\LESSON                         PRG: LESSON                 CONFIG VALID
REPLACE
```

The Print Configuration menu defines which parts of the rack configuration should be printed out and the destination of the listing.

To save the content of this screen to a file called PSCREENS.SET, press **Save** (**F10**). This file is located in the directory of the active program folder and is read when the print function is entered. If you do not save the content of this screen, the default values will be used the next time you print program logic, cross references, variables, or program annotation.

Refer to the following definitions when changing the entries on the Print Configuration screen.

Table 9-5. Print Configuration Parameters

Field	Description
Title	An optional title of up to 62 characters centered at the top of each page.
Subtitle	An optional line of up to 62 characters printed below the title on each page.
I/O Rack	To print rack information, enter Y (Yes).
Detail	To print the detailed configuration screens, enter Y (Yes). This field is only available when the selection for the ***I/O Rack*** field is Y.
Reference View	To print the reference view tables, enter Y (Yes).
CPU Configuration	To print CPU configuration data, enter Y (Yes).
Starting Page Number	To begin numbering the printout with a page number other than 1, enter the number here. The value may range from 1 to +32,767.
From I/O Rack	Enter the number of the first rack whose configuration data is to be printed. The value of this field may not exceed the value of the ***To I/O Rack*** field.
To I/O Rack	Enter the number of the last rack whose configuration data is to be printed. The value of this field may not be less than the value of the ***From I/O Rack*** field. In order to print a single rack of information, both the ***From I/O Rack*** and ***To I/O Rack*** fields must contain the same value.
Port	Specify the destination of the listing.
File Name	If "file" is selected for the destination in the ***Port*** field, enter the name of the file to which the printout is to be directed. Any valid file specification may be entered in this field. A new file is always created, overwriting any existing file of the same name.

Pagination Guidelines

Only one type of screen can be printed on a given page, but multiple detail screens may be printed on a single page if they will fit on that page following these guidelines:

Table 9-6. Pagination Guidelines

Field	Description
Rack	The printing of a given rack cannot be broken across pages. Only one rack per page is permitted.
Detail	The printing of a given detail screen cannot be broken across pages. Because the length of a listing page may be changed and there are a variable number of detail screens for a given module, all the screens of a particular module may not be displayed on the same page.

Section 7: Print Function Examples

The following examples illustrate various hard copy listings which can be produced with the print function.

Certain information is presented on every page, regardless of what specific item is currently being printed. The first line of each page of the listing will contain the date, time, software version number, and page number. The user-supplied title and subtitle are centered on the second and third lines of the page, respectively. The program name appears in the bottom left corner of the page. The current program folder specification appears in the bottom center of the page.

The header page is printed when Y (Yes) is selected for the header page option on the Print Program Logic screen. The header page for the program includes the program name, the configured reference sizes supported by the host PLC, the highest reference address used in the program, and the size of the program in bytes. This is an example program header page:

```
 01-15-93  12:02 GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v4.00)      Page   1



                  GGGG EEEEE       FFFFF  AAA  N   N U   U  CCCC
                 G     E           F     A   A NN  N U   U C
                 G GGG EEEE        FFF   AAAAA N N N U   U C
                 G   G E           F     A   A N  NN U   U C
                  GGG  EEEEE       F     A   A N   N  UUU   CCCC



            AAA  U   U TTTTT  OOO  M   M  AAA  TTTTT IIIII  OOO  N   N
           A   A U   U   T   O   O MM MM A   A   T     I   O   O NN  N
           AAAAA U   U   T   O   O M M M AAAAA   T     I   O   O N N N
           A   A U   U   T   O   O M   M A   A   T     I   O   O N  NN
           A   A  UUU    T    OOO  M   M A   A   T   IIIII  OOO  N   N



(******************************************************************************)
(*                                                                            *)
(*                                Program:  NEW                               *)
(*                                                                            *)
(*        PLC PROGRAM ENVIRONMENT               HIGHEST REFERENCE USED        *)
(*       --------------------------           --------------------------      *)
(*                INPUT (%I):   512                     INPUT: %I0100         *)
(*               OUTPUT (%Q):   512                    OUTPUT: %Q0061         *)
(*             INTERNAL (%M):  1024                  INTERNAL: %M0006         *)
(*          GLOBAL DATA (%G):  1280               GLOBAL DATA:  NONE          *)
(*            TEMPORARY (%T):   256                 TEMPORARY:  NONE          *)
(*             REGISTER (%R):  2048                  REGISTER: %R0003         *)
(*        ANALOG INPUT (%AI):   128              ANALOG INPUT:  NONE          *)
(*       ANALOG OUTPUT (%AQ):    64             ANALOG OUTPUT:  NONE          *)
(*                                                                            *)
(*                     PROGRAM SIZE (BYTES):       208                        *)
(*                                                                            *)
(*                                                                            *)
(******************************************************************************)










 Program: NEW                  D:\ACME\CONVEYOR\NEW
```

This example printout was generated by selecting only **Y** (Yes) for the IL Logic option.

```
01-15-93  12:02 GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v4.00)     Page   1


  << START OF BLOCK _MAIN >>

  << RUNG 4   STEP #0001 >>

      #0001  01   NOOP

  << RUNG 5   STEP #0002 >>

      #0002  LD        %I0004
      #0003  AND       %Q0002
      #0004  AND       %Q0044
      #0005  AND       %Q0012
      #0006  OR   NOT  %Q0005
      #0007  AND  NOT  %I0003
      #0008  AND  NOT  %Q0025
      #0009  AND  NOT  %Q0006
      #0010  AND       %Q0061
      #0011  OUT       %Q0005

  << RUNG 6   STEP #0012 >>

      #0012  LD        %I0002
      #0013  LD        %I0100
      #0014  OR   NOT  %Q0025
      #0015  AND  BLK
      #0016  LD        %Q0020
      #0017  OR   NOT  %Q0004
      #0018  AND  BLK
      #0019  AND       %I0073
      #0020  OUT       %Q0025

  << RUNG 7   STEP #0021 >>

      #0021  LD   NOT  %Q0005
      #0022  AND  NOT  %Q0012
      #0023  LD   NOT  %M0005
      #0024  AND       %M0006
      #0025  OR   BLK
      #0026  AND       %I0021
      #0027  AND       %I0022
      #0028  OUT       %M0001

  << RUNG 8   STEP #0029 >>

      #0029  LD   NOT  %Q0044
      #0030  AND       %M0002
      #0031  FUNC 60   ADD
                  P1:  %R0001
                  P2:  %R0002
                  P3:  %R0003
      #0032  OUT       %Q0006

      #0033  END OF PROGRAM



Program: NEW                   D:\ACME\CONVEYOR\NEW                 Block: _MAIN
```

In this example printout, Logic is set to **Y** (Yes), and all Logic options (Reference List, Nickname and Reference, Reference Description, and Rung Comments) are set to **N** (No).

```
01-15-93  12:02 GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v4.00)     Page     1
                                TITLE APPEARS HERE
                              SUBTITLE APPEARS HERE


|[  START OF LD  PROGRAM SAMPLE   ]        (* This is a sample program         *)
|
|[     VARIABLE DECLARATIONS      ]
|
|[       BLOCK DECLARATIONS       ]
|
|[     START OF PROGRAM LOGIC     ]
|
| << RUNG 4  STEP #0001 >>
|
|I140_00 I141_07 SMP_PMP                                                   %T0086
+--] [--+--] [--+--]/[--+---------------------------------------------------(S)--
|       |       |       |
|       |B152_00|       |                                                 B152_00
|       +--] [--+       +---------------------------------------------------(S)--
|
| << RUNG 5  STEP #0008 >>
|
|I140_00 I141_07                 +-----+                                  SMP_PMP
+--] [--+--] [--+----------------+ONDTR+------------------------------------( )--
|       |       |                |0.10s|
|       |B152_00|                |     |
|       +--] [--+        +-------+R    |
|                        |       |     |
|I140_01                 |       |     |
+--] [-------------------+ CONST -+PV   |
|                          +00100 |     |
|                                 +-----+
|                                 %R0004
|
|(*  COMMENT  *)
|
| << RUNG 7  STEP #0016 >>
|
|I141_01                                                                  B152_00
+--]/[----------------------------------------------------------------------(R)--
|
| << RUNG 8  STEP #0018 >>
|
|PB_SUM                                                                    %T0075
+--] [--+-------------------------------------------------------------------( )--
|       |
|BAD_RAM|                                                                  %T0107
+--] [--+-------------------------------------------------------------------( )--
|
|[      END OF PROGRAM LOGIC      ]
|



  Program: SAMPLE                 D:\30FOLDERS\SAMPLE                Block: _MAIN
```

In this example printout, the Nickname and Reference option is set to **Y** (Yes) and the Reference Description option is set to **N** (No). Both a nickname, if defined, and a reference address are printed on an instruction operand.

```
01-15-93  12:02 GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v4.00)     Page     1
                                  TITLE APPEARS HERE
                                SUBTITLE APPEARS HERE


|[  START OF LD  PROGRAM SAMPLE    ]       (* This is a sample program        *)
|
|[     VARIABLE DECLARATIONS       ]
|
|[       BLOCK DECLARATIONS        ]
|
|[     START OF PROGRAM LOGIC      ]
|
| << RUNG 4  STEP #0001 >>
|
|I140_00 I141_07 SMP_PMP
|%I0501  %I0505  %M0997                                                   %T0086
+--] [--+--] [--+--] [--+-----------------------------------------------------(S)--
|       |       |       |
|       |B152_00|       |                                                 B152_00
|       |%M0627 |       |                                                  %M0627
|       +--] [--+       +-----------------------------------------------------(S)--
|
| << RUNG 5  STEP #0008 >>
|
|I140_00 I141_07                                                         SMP_PMP
|%I0501  %I0505                +------+                                   %M0997
+--] [--+--] [--+--------------+ONDTR+---------------------------------------( )--
|       |       |              |0.10s|
|       |B152_00|              |     |
|       |%M0627 |              |     |
|       +--] [--+       +------+R    |
|                       |      |     |
| 140_01                |      |     |
|%I0502                 |      |     |
+--] [------------------+ CONST -+PV  |
|                         +00100 |     |
|                              +------+
|
|                              %R0004
|
|(*  COMMENT  *)
|
| << RUNG 7  STEP #0016 >>
|
|I141_01                                                                 B152_00
|%I0504                                                                   %M0627
+--]/[----------------------------------------------------------------------(R)--
|
| << RUNG 8  STEP #0018 >>
|
|PB_SUM
|%SA001                                                                   %T0075
+--] [--+-------------------------------------------------------------------( )--
|       |
|BAD_RAM|
|%SB010 |                                                                 %T0107
+--] [--+-------------------------------------------------------------------( )--
|


 Program: SAMPLE               D:\30FOLDERS\SAMPLE                  Block: _MAIN
```

In this example printout, the Nickname and Reference option is set to **N** (No) and the Reference Description option is set to **Y** (Yes). With the Reference Description option enabled, a four-line reference description, if defined, is printed above each instruction operand. Since the Nickname and Reference option is not selected, only a single line of either a nickname or a reference address is printed on an instruction operand.

```
 01-15-93  12:02 GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v4.00)     Page   1
                                 TITLE APPEARS HERE
                               SUBTITLE APPEARS HERE

|[  START OF LD  PROGRAM SAMPLE   ]       (* This is a sample program         *)
|
|[     VARIABLE DECLARATIONS      ]
|
|[       BLOCK DECLARATIONS       ]
|
|[     START OF PROGRAM LOGIC     ]
|
| << RUNG 4  STEP #0001 >>
|
|        Intake  Sump
|        Valve   Pump 1
|        Control Density                                                  Conveyr
|        Switch  meter                                                    Check
|I140_00 I141_07 SMP_PMP                                                  %T0086
+--] [--+--] [--+--]¯[--+--------------------------------------------------(S)--
|       |       |       |
|       |Bulb   |       |                                                 Bulb
|       |B152   |       |                                                 B152
|       |Circ 00|       |                                                 Circ 00
|       |Switch |       |                                                 Switch
|       |B152_00|       |                                                 B152_00
|       +--] [--+       +--------------------------------------------------(S)--
|
| << RUNG 5  STEP #0008 >>
|
|        Intake                                                           Sump
|        Valve                                                            Pump 1
|        Control                                                          Density
|        Switch                                                           meter
|I140_00 I141_07                  +------+                                SMP_PMP
+--] [--+--] [--+-----------------+ONDTR+----------------------------------(¯)--
|       |       |                 |0.10s|
|       |Bulb   |                 |      |
|       |B152   |                 |      |
|       |Circ 00|                 |      |
|       |Switch |                 |      |
|       |B152_00|                 |      |
|       +--] [--+       +---------+R     |
|                       |         |      |
|Air                    |         |      |
|Intake                 |         |      |
|Valve                  |         |      |
|Switch                 |         |      |
|I140_01                |         |      |
+--] [------------------+ CONST -+PV     |
|                        +00100  |      |
|                                +------+
|                                Seconds
|                                to chk
|                                complet
|                                ion
|                                %R0004


 Program: SAMPLE               D:\30FOLDERS\SAMPLE                  Block: _MAIN
```

This example printout was generated by selecting **Y** (Yes) for the Cross Reference Table and Implicit Cross Reference options, and for %M on the Select Cross Reference Options screen.

```
01-15-93  12:02 GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v4.00)      Page  1


                      ***** I N T E R N A L *****
              ***** C R O S S   R E F E R E N C E   T A B L E *****

 REFERENCE    NICKNAME    REFERENCE DESCRIPTION / CROSS REFERENCES
 ---------    --------    --------- ----------- - ----- ----------

 %M0247     : MSTP_14   ::STEP 14 MAIN BIT SEQ
                          -] [- 287
                          FBIO  (296)

 %M0248     : MSTP_15   ::STEP 15 MAIN BIT SEQ
                          -] [- 287, 294, 295, 512
                          FBIO  (296)

 %M0249     : MSTP_16   ::STEP 16 MAIN BIT SEQ
                          -] [- 287, 293, 480, 481, 482, 483, 508, 513, 514, 516,
                                519, 520, 521, 523, 526, 527, 528, 530, 533, 534,
                               535, 537, 540, 553
                          FBIO  (296)

 %M0250     : MSTP_17   ::STEP 17 MAIN BIT SEQ
                          -] [- 227, 228, 229, 288, 293, 591, 592
                          FBIO  (296)

 %M0251     : MSTP_18   ::STEP 18 MAIN BIT SEQ
                          -] [- 262, 263, 288, 291
                          FBIO  (296)

 %M0252     : MSTP_19   ::STEP 19 MAIN BIT SEQ
                          -] [- 232, 265, 288, 487, 641
                          FBIO  (296)

 %M0253     : MSTP_20   ::STEP 20 MAIN BIT SEQ
                          -] [- 275, 289, 487
                          FBIO  (296)

 %M0254     : MSTP_21   ::STEP 21 MAIN BIT SEQ
                          -] [- 289, 588
                          FBIO  (296)

 %M0255     : MSTP_22   ::STEP 22 MAIN BIT SEQ
                          -] [- 267, 269, 271, 273, 289, 590
                          -]/[- 268, 270, 272, 274
                          FBIO  (296)

 %M0256     : MSTP_23   ::STEP 23 MAIN BIT SEQ
                          -] [- 156, 289
                          FBIO  (296)

 %M0257     : MSTP_24   ::STEP 24 MAIN BIT SEQ
                          -] [- 154, 289
                          FBIO  (296)



 Program: SAMPLE                 D:\30FOLDERS\SAMPLE                Block: _MAIN
```

This example printout was generated by selecting **Y** (Yes) for the Use Tables and Implicit Cross Reference options, and for %M on the Select Cross Reference Options screen.

```
01-15-93  12:02 GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v4.00)      Page    1



                  *****   I N T E R N A L   U S E   T A B L E   *****


REF.    |64      |56      |48      |40      |32      |24      |16      |8       |1
ADDRESS +------- +------- +------- +------- +------- +------- +------- +-------+

%M0064  ******** *---**** ******** ******** ******** ******** ******** ********
%M0128  ******** ****---* ******** ******** ******-- -------* --****** **----**
%M0192  ******** ***-**-- ------** ******** ******** **-**-** -**-**** ********
%M0256  ######## ######## #######* ******** ***-**** ******** ******** ----****

%M0320  -------- -------- +++####* +++++### #######* +++++### #######* ++++####
%M0384  ******** ******** ******** ++++#### #######* ++++++++ ++#####* ******-*
%M0448  ----**** ******** ******** ******** ******** --**-*** ******** ********
%M0512  ++#####* ++++++++ ++++#### ######## #######* -----*** ******** ********

%M0576  -------- -------- -------- ++++++++ ++++++## ######## #######* ++++++++
%M0640  -------- -------- -------- -------- -------- -------- -------- --------
%M0704  -------- -------- -------- -------- -------- -------- -------- --------
%M0768  -------- -------- -------- -------- -------- -------- -------- --------

%M0832  -------- -------- -------- -------- -------- -------- -------- --------
%M0896  -------- -------- -------- -------- -------- -------- -------- --------
%M0960  -------- -------- -------- -------- -------- -------- -------- --------
%M1024                             **------ -------- -------- -------- --------

- no use      * explicit use      + implicit use      # explicit and implicit use




Program: SAMPLE                   D:\30FOLDERS\SAMPLE                 Block: _MAIN
```

This example printout was generated by selecting **Y** (Yes) for the Value Tables option and setting the %M reference address range to 1 to 1024 on the Print Values screen.

```
01-15-93  12:02 GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v4.00)      Page    1


               *****  I N T E R N A L   V A L U E   T A B L E   *****


 REF.    |64      |56      |48      |40      |32      |24      |16      |8      |1
 ADDRESS +-------- +-------- +-------- +-------- +-------- +-------- +-------- +-------+

 %M0064  00000000 00000100 00000000 00000000 00000000 00011000 00000000 00010010
               *****  I N T E R N A L   V A L U E   T A B L E   *****


 REF.    |64      |56      |48      |40      |32      |24      |16      |8      |1
 ADDRESS +-------- +-------- +-------- +-------- +-------- +-------- +-------- +-------+

 %M0064  00000000 00000100 00000000 00000000 00000000 00011000 00000000 00010010
 %M0128  00000000 00000011 10010100 00000000 00000000 00000000 00001100 00000000
 %M0192  00000000 00000100 00000110 00000000 00000000              0008 00000000

 %M0256  00000000 00000100 00000000 00000000 00000000 00100101 01110011 00001111
 %M0320             +00724 00000001 10000000 01000111 01010000 01000000 10000000
 %M0384  00000000 00000100 00000000 01000000 00000001 00010000 01000001 10010000

 %M0448  00000000 00000000 00000000              6179 00000001 11011100 10000100
 %M0512  00000001 00000100 00000000 00000000 00000001 01010010 00000000 00000100
 %M0576  00000000 00000000 00001111 00000000 00000000 00000010 00010001 00000000

 %M0640  00000000 00000000 00000000 00000000 00000000            +09232 00000100
 %M0704  00000000 00000000 00000000 00000000 00000000 00000100 10101100 10011000
 %M0768  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

 There are no non-zero values in the range from %M0769 to %M1024.




 Program: SAMPLE                 D:\30FOLDERS\SAMPLE
```

This example printout was generated by selecting **Y** (Yes) for the Overrides option and setting the %M reference address range to 1 to 1024 on the Print Values screen.

```
01-15-93  12:02 GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v4.00)     Page    1


          *****  I N T E R N A L   O V E R R I D E   T A B L E   *****


 REF.    |64      |56      |48      |40      |32      |24      |16      |8      |1
 ADDRESS +--------+--------+--------+--------+--------+--------+--------+-------+

 %M0064  -------- -------- -------- -------- -------- -------- -------- --------
 %M0128  -------- -------- ------0- -------- -------- -------- -------- --------
 %M0192  -------- ----01-- -------- -------- -------- -------- -------- --------

 %M0256  -------- ---0-1-- -------- -----0-- -------- -------- -------- --------
 %M0320  -------- -----1-- -------- -------- -------- -------- -------- --------
 %M0384  -------- ------00 0------- -------0 -------- ---1---- -------- --------

 %M0448  -------- -------- -------- -------- -------- 0------- -------- --------
 %M0512  -------- -------- 00000000 000--0-- ----0--- -10-0--- -------- --------
 %M0576  -------- ---0---- -------- -------- -------- -------- -------- --------

 %M0640  -------- -------- -------- -------- -------- -------- -------- --------
 %M0704  -------- -------- ---0---- -------- -------- -------- -------- --------
 %M0768  -------- ---0--0- -------- -------- -------- -------- -------- --------

 %M0832  -------- -------- -------- -------- -------- -------- -------- --------
 %M0896  -------- -------- -------- -------- -------- -------- -------- --------
 %M0960  -------- -------- -------- -------- -------- -------- -------- --------

 %M1024  -------- -------- -------- -------- -------- -------- -------- --------

 - = not overridden    0 = overridden in OFF state    1 = overridden in ON state











 Program: SAMPLE              D:\30FOLDERS\SAMPLE
```

This example printout of an 80-column listing of a short program was generated by selecting **Y** (Yes) for the IL Logic, Logic, and In Ladder Cross Reference options.

```
 01-15-93  12:02 GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v4.00)     Page    1
                                 TITLE APPEARS HERE
                               SUBTITLE APPEARS HERE


|[  START OF LD  PROGRAM   NEW    ]       (*                                  *)
|
|[     VARIABLE DECLARATIONS      ]
|
|[       BLOCK DECLARATIONS       ]
|
|[     START OF PROGRAM LOGIC     ]
|
|(*  COMMENT  *)
|     #0001  01   NOOP
|
| << RUNG 5  STEP #0002 >>    Cross reference for %Q0005
|                             -]/[- 5, 7
|
|%I0004  %Q0002  %Q0044  %Q0012  %I0003  %Q0025  %Q0006  %Q0061          %Q0005
+--] [-----] [-----] [-----] [--+--]/[-----]/[-----]/[-----] [--------------( )--
|                               |         0006    0008
|%Q0005                         |
+--]/[--------------------------+
| 0005
|
|     #0002  LD        %I0004
|     #0003  AND       %Q0002
|     #0004  AND       %Q0044
|     #0005  AND       %Q0012
|     #0006  OR   NOT  %Q0005
|     #0007  AND  NOT  %I0003
|     #0008  AND  NOT  %Q0025
|     #0009  AND  NOT  %Q0006
|     #0010  AND       %Q0061
|     #0011  OUT       %Q0005
|
| << RUNG 6  STEP #0012 >>    Cross reference for %Q0025
|                             -]/[- 5, 6
|
|%I0002  %I0100  %Q0020  %I0073                                          %Q0025
+--] [--+--] [--+--] [--+--] [----------------------------------------------( )--
|       |       |       |
|       |%Q0025 |%Q0004 |
|       +--]/[--+--]/[--+
|         0006
|
|     #0012  LD        %I0002
|     #0013  LD        %I0100
|     #0014  OR   NOT  %Q0025
|     #0015  AND  BLK
|     #0016  LD        %Q0020
|     #0017  OR   NOT  %Q0004
|     #0018  AND  BLK
|     #0019  AND       %I0073
|     #0020  OUT       %Q0025
|



  Program: NEW                   D:\ACME\CONVEYOR\NEW                Block: _MAIN
```

```
01-15-93  12:02 GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v4.00)     Page    2
                                   TITLE APPEARS HERE
                                  SUBTITLE APPEARS HERE


|<< RUNG 7  STEP #0021 >>    Cross reference for %M0001
|                                      NONE
|
|%Q0005  %Q0012  %I0021  %I0022                                              %M0001
+--]/[-----]/[--+--] [-----] [----------------------------------------------( )--
| 0005          |
|%M0005  %M0006 |
+--]/[-----] [--+
|
|     #0021  LD   NOT  %Q0005
|     #0022  AND  NOT  %Q0012
|     #0023  LD   NOT  %M0005
|     #0024  AND       %M0006
|     #0025  OR   BLK
|     #0026  AND       %I0021
|     #0027  AND       %I0022
|     #0028  OUT       %M0001
|
| << RUNG 8  STEP #0029 >>    Cross reference for %Q0006
|                             -]/[- 5
|
|%Q0044  %M0002  +------+                                                    %Q0006
+--]/[-----] [---+ ADD  +---------------------------------------------------( )--
|                | INT  |
|                |      |
|       %R0001 --+I1   Q+--%R0003
|                |      |
|       %R0002 --+I2    |
|                +------+
|
|     #0029  LD   NOT  %Q0044
|     #0030  AND       %M0002
|     #0031  FUNC 60   ADD
|                 P1:  %R0001
|                 P2:  %R0002
|                 P3:  %R0003
|     #0032  OUT       %Q0006
|
|[      END OF PROGRAM LOGIC       ]
|
|     #0033  END OF PROGRAM
|








 Program: NEW                  D:\ACME\CONVEYOR\NEW                   Block: _MAIN
```

This example printout of an 80-column listing was generated by selecting **Y** (Yes) for the IL Logic, Logic, Reference List, Rung Comments, and In Ladder Cross Reference options.

```
01-15-93   12:02 GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v4.00)      Page     1
                                    TITLE APPEARS HERE
                                  SUBTITLE APPEARS HERE

|[  START OF LD  PROGRAM NEWPROG  ]        (*                                   *)
|
|[      VARIABLE DECLARATIONS      ]
|
|[        BLOCK DECLARATIONS       ]
|
|[      START OF PROGRAM LOGIC     ]
|
| (****************************************************************************)
| (* The following logic rung enables automatic mode when all enabling condi-  *)
| (* tions are met.  See operator's manual, page 6-12, for more information.   *)
| (****************************************************************************)
|     #0001  01    NOOP
|
| << RUNG 5  STEP #0002 >>     Cross reference for AUTO
|                              -]/[- 5, 7
|
|AUTOPB   LISUP  UNISUP  %Q0012  RSTAHPB EMERGST %Q0006  COOLANT            AUTO
+--] [-----] [-----] [-----] [--+--]/[-----]/[-----]/[-----] [--------------( )--
|                               |          0006    0008
| AUTO                          |
+--]/[--------------------------+
| 0005
|
|     #0002  LD         %I0004
|     #0003  AND        %Q0002
|     #0004  AND        %Q0044
|     #0005  AND        %Q0012
|     #0006  OR    NOT  %Q0005
|     #0007  AND   NOT  %I0003
|     #0008  AND   NOT  %Q0025
|     #0009  AND   NOT  %Q0006
|     #0010  AND        %Q0061
|     #0011  OUT        %Q0005
|



 REFERENCE NICKNAME      REFERENCE DESCRIPTION
  %Q0006                 Hand Light
  %Q0012                 Table is out
  %Q0005   AUTO          Auto Light
  %I0004   AUTOPB        Auto Push Button
  %Q0061   COOLANT       Coolant Valve
  %Q0025   EMERGST       Emergency Return
  %Q0002   LISUP         Loader is up
  %I0003   RSTAHPB       Reset Auto / Hand PB
  %Q0044   UNISUP        Unloader is up

 Program: NEWPROG            D:\ACME\CONVEYOR\NEWPROG                Block: _MAIN
```

```
01-15-93  12:02 GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v3.00)     Page   2
                                       TITLE APPEARS HERE
                                     SUBTITLE APPEARS HERE

| << RUNG 6  STEP #0012 >>     Cross reference for EMERGST
|                              -]/[- 5, 6
|
|EMSTOP   START  CLAMPED %I0073                                          EMERGST
+--] [--+--] [--+--] [--+--] [-------------------------------------------( )--
|       |       |       |
|       |EMERGST|SPN MTR|
|       +--]/[--+--]/[--+
|         0006
|
|     #0012  LD         %I0002
|     #0013  LD         %I0100
|     #0014  OR    NOT  %Q0025
|     #0015  AND   BLK
|     #0016  LD         %Q0020
|     #0017  OR    NOT  %Q0004
|     #0018  AND   BLK
|     #0019  AND        %I0073
|     #0020  OUT        %Q0025
|
| << RUNG 7  STEP #0021 >>     Cross reference for %M0001
|                                        NONE
|
| AUTO   %Q0012                                                          %M0001
+--]/[-----]/[-----------------------------------------------------------( )--
| 0005
|
|     #0021  LD    NOT  %Q0005
|     #0022  AND   NOT  %Q0012
|     #0023  OUT        %M0001
|


 REFERENCE NICKNAME       REFERENCE DESCRIPTION
  %I0073
  %Q0012                  Table is out
  %M0001
  %Q0005   AUTO           Auto Light
  %Q0020   CLAMPED
  %Q0025   EMERGST        Emergency Return
  %I0002   EMSTOP         Emergency Stop PB
  %Q0004   SPN MTR        Start Spindle Motor
  %I0100   START

 Program: NEWPROG          D:\ACME\CONVEYOR\NEWPROG              Block: _MAIN
```

```
                 GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v3.00)      Page      3
                                 TITLE APPEARS HERE
                               SUBTITLE APPEARS HERE


| << RUNG 8  STEP #0024 >>     Cross reference for %Q0006
|                              -]/[- 5
|
|UNISUP  %M0002  +------+                                                    %Q0006
+--]/[-----] [---+ ADD_ +----------------------------------------------------( )--
|                | INT  |
|                |      |
|        %I0097 -+I1   Q+-%Q0097
|                |      |
|        %I0113 -+I2    |
|                +------+
|
|     #0024  LD    NOT   %Q0044
|     #0025  AND         %M0002
|     #0026  FUNC  60    ADD
|                  P1:   %I0097
|                  P2:   %I0113
|                  P3:   %Q0097
|     #0027  OUT         %Q0006
|
|[       END OF PROGRAM LOGIC      ]
|
|     #0028  END OF PROGRAM
|





  REFERENCE NICKNAME      REFERENCE DESCRIPTION
   %I0097
   %I0113
   %Q0006                 Hand Light
   %Q0097
   %M0002
   %Q0044   UNISUP        Unloader is up
  Program: NEWPROG            D:\ACME\CONVEYOR\NEWPROG              Block: _MAIN
```

This example printout of a 132-column listing was generated by selecting **Y** (Yes) for the Logic, Rung Comments, Nicknames and References, Reference List, and In Ladder Cross Reference options.

```
01-15-93     15:38              GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v4.00)          page 1
                                             TITLE APPEARS HERE
                                            SUBTITLE APPEARS HERE

|[  START OF LD  PROGRAM NEWPROG  ]      (*                                  *)
|
|[     VARIABLE DECLARATIONS      ]
|
|[       BLOCK DECLARATIONS       ]
|
|[     START OF PROGRAM LOGIC     ]
|
|**************************************************************************)
| (* The following logic rung enables automatic mode when all enabling condi- *)
| (* tions are met.  See operator's manual, page 6-12, for more information.  *)
|**************************************************************************)

|AUTOPB   LISUP  UNISUP  %Q0012  RSTAHPB EMERGST %Q0006  COOLANT         AUTO    << RUNG 5   STEP #0002 >>
|%I0004  %Q0002  %Q0044  %Q0012  %I0003  %Q0025  %Q0006  %Q0061        %Q0005
+--] [-----] [-----] [-----] [--+--]/[-----]/[-----]/[-----] [-------------( )--  Cross reference for AUTO
|                               |               0006    0008                       -]/[-   5,  7
| AUTO                          |
+--]/[--------------------------+
| 0005
|
|EMSTOP   START  CLAMPED %I0073                                          EMERGST   << RUNG 6   STEP #0012 >>
+--] [--+--] [--+--] [--+--] [--------------------------------------------( )-- Cross reference for AUTO
|       |       |       |                                                          -]/[-   5,  7
|       |EMERGST|SPN_MTR|
|       +--]/[--+--]7[--+
|         0006
|
| AUTO                                                                            << RUNG 7   STEP #0021 >>
| %Q0005     %Q0012                                                      %M0001
+--]/[-----]/[------------------------------------------------------------( )--  Cross reference for %M0001
| 0005                                                                                       NONE
|

 REFERENCE NICKNAME      REFERENCE DESCRIPTION           REFERENCE NICKNAME      REFERENCE DESCRIPTION
  %I0073                                                     %Q0025  EMERGST        Emergency Return
  %Q0006                       Hand Light                    %I0002  EMSTOP         Emergency Stop PB
  %Q0012                       Table is out                  %Q0002  LISUP          Loader is up
  %M0001                                                     %I0003  RSTAHPB        Reset Auto / Hand PB
  %Q0005    AUTO               Auto Light                    %Q0004  SPN_MTR        Start Spindle Motor
  %I0004    AUTOPB             Auto Push                     %I0100  START
  %Q0020    CLAMPED                                          %Q0044  UNISUP         Unloader is up
  %Q0061    COOLANT            Coolant Valve

 Program: NEWPROG                                      C:\ACME\CONVEYOR\NEWPROG
```

```
01-15-92     15:38             GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v4.00)          page 2
                                              TITLE APPEARS HERE
                                             SUBTITLE APPEARS HERE

| UNISUP
| %Q0044 %M0002  +------+                                                 %Q0006
|--]/[-----] [---+ ADD  +-------------------------------------------------( )--
|                | INT  |
|                |      |
|        %I0097 -+I1   Q+-%Q0097
|                |      |
|        %I0113 -+I2    |
|                +------+
|
|
|[      END OF PROGRAM LOGIC        ]
|

  REFERENCE NICKNAME    REFERENCE DESCRIPTION    REFERENCE NICKNAME REFERENCE DESCRIPTION
   %I0097                                            %Q0097
   %I0113                                            %M0002
   %Q0006                  Hand Light               %Q0044    UNISUP      Unloader is up

 Program: NEWPROG                      C:\ACME\CONVEYOR\NEWPROG
```

The print function must be able to print large rungs which have been truncated by the editor. Truncated rungs will be shown in the listing as they appear on the editor screen. If the number of allowed columns is exceeded, the special truncation character (+) is printed in the rightmost column. If the number of allowed rows is exceeded, the special truncation character (+) is printed below the seventh rung line (bottom displayable row).

The following examples show truncated rungs which exceed both the displayable width and height.

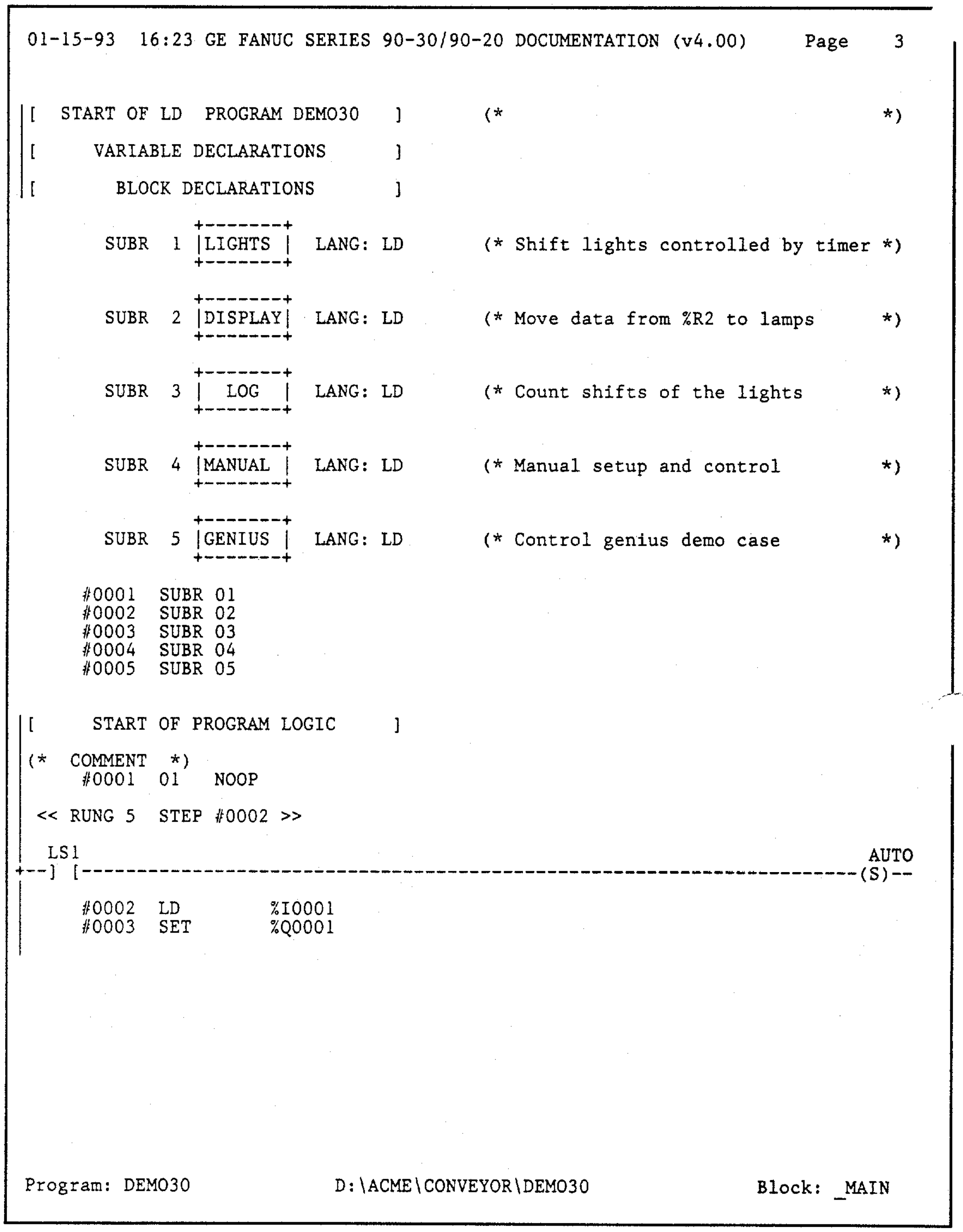
```
 01-15-93     12:02 GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v4.00)      Page   1
                                   TITLE APPEARS HERE
                                 SUBTITLE APPEARS HERE


|[  START OF LD  PROGRAM  TEMP     ]       (*                                    *)
|
|[     VARIABLE DECLARATIONS       ]
|
|[       BLOCK DECLARATIONS        ]
|
|[     START OF PROGRAM LOGIC      ]
|
| (*****************************************************************************)
| (* This is a sample program to help illustrate the print feature.            *)
| (* The following rung is truncated on the right and on the bottom.          *)
| (*****************************************************************************)
|
| << RUNG 5  STEP #0002 >>
|
|%I0001  %M0001  %M0002  %M0003  %M0004  %M0005  %M0006  %M0007  %M0008
+--] [--+--] [-----] [-----] [-----] [-----] [-----] [-----] [-----] [---+
|       |
|%I0002 |
+--] [--+                                                                +
|       |
|%I0003 |
+--] [--+                                                                +
|       |
|%I0004 |
+--] [--+                                                                +
|       |
|%I0005 |
+--] [--+                                                                +
|       |
|%I0006 |
+--] [--+                                                                +
|       |
|%I0007 |
+--] [--+                                                                +
|       |
|+      |+      +       +       +       +       +       +       +       +
|




 Program: TEMP                           A:\TEMP
```

```
01-15-93     12:02 GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v4.00)      Page   2
                                  TITLE APPEARS HERE
                                SUBTITLE APPEARS HERE


|(*****************************************************************************)
|(* The following rung is truncated on the right only.                        *)
|(*****************************************************************************)
|
|<< RUNG 7  STEP #0023 >>
|
|%T0001  +------+                  +------+                  +------+
+--] [---+ ADD  +------------------+ DIV  +------------------+MOVE  +----------+
|        | INT‾ |                  | INT‾ |                  | INT‾ |
|        |      |                  |      |                  |      |
|%R0001 -+I1   Q+-%R0003  %R0004 -+I1   Q+-%R0006  %R0007 -+IN   Q+-%R0008  +
|        |      |                  |      |                  | LEN  |
|        |      |                  |      |                  |00001 |
|%R0002 -+I2    |         %R0005 -+I2    |                  +------+          +
|        +------+                  +------+
|
|[      END OF PROGRAM LOGIC      ]
|




Program: TEMP                        A:\TEMP
```

The following pages are sample pages from a program named DEMO30, which calls several subroutines. One of these subroutines, named LIGHTS, is also included in this sample printout.

```
01-15-93  16:23 GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v4.00)     Page    1



                 GGGG EEEEE      FFFFF  AAA  N   N U   U  CCCC
                G     E          F     A   A NN  N U   U C
                G GGG EEEE       FFF   AAAAA N N N U   U C
                G   G E          F     A   A N  NN U   U C
                 GGG  EEEEE      F     A   A N   N  UUU   CCCC


           AAA  U   U TTTTT  OOO  M   M  AAA  TTTTT IIIII  OOO  N   N
          A   A U   U   T   O   O MM MM A   A   T     I   O   O NN  N
          AAAAA U   U   T   O   O M M M AAAAA   T     I   O   O N N N
          A   A U   U   T   O   O M   M A   A   T     I   O   O N  NN
          A   A  UUU    T    OOO  M   M A   A   T   IIIII  OOO  N   N


(******************************************************************************)
(*                                                                            *)
(*                           Program:  DEMO30                                 *)
(*                                                                            *)
(*        PLC PROGRAM ENVIRONMENT               HIGHEST REFERENCE USED        *)
(*       -------------------------            --------------------------      *)
(*                INPUT (%I):   512                      INPUT:  %I0200       *)
(*               OUTPUT (%Q):   512                     OUTPUT:  %Q0010       *)
(*             INTERNAL (%M):  1024                   INTERNAL:  %M0001       *)
(*          GLOBAL DATA (%G):  1280                GLOBAL DATA:     NONE      *)
(*            TEMPORARY (%T):   256                  TEMPORARY:     NONE      *)
(*             REGISTER (%R):  2048                   REGISTER:  %R0002       *)
(*        ANALOG INPUT (%AI):   128               ANALOG INPUT:     NONE      *)
(*       ANALOG OUTPUT (%AQ):    64              ANALOG OUTPUT:     NONE      *)
(*                                                                            *)
(*                    PROGRAM SIZE (BYTES):      848                          *)
(*                                                                            *)
(*                                                                            *)
(******************************************************************************)










Program: DEMO30              D:\ACME\CONVEYOR\DEMO30
```

```
01-15-93  16:23 GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v4.00)      Page   2









(****************************************************************************)
(*                                                                          *)
(*                              BLOCK:  _MAIN                               *)
(*                                                                          *)
(*                                                                          *)
(*                            BLOCK SIZE (BYTES):   170                     *)
(*                       DECLARATIONS (ENTRIES):     9                      *)
(*                                                                          *)
(*                                                                          *)
(*                         HIGHEST REFERENCE USED                           *)
(*                      -----------------------------                       *)
(*                                                                          *)
(*                              INPUT (%I):  %I0002                         *)
(*                             OUTPUT (%Q):  %Q0001                         *)
(*                           INTERNAL (%M):     NONE                        *)
(*                        GLOBAL DATA (%G):     NONE                        *)
(*                          TEMPORARY (%T):     NONE                        *)
(*                           REGISTER (%R):     NONE                        *)
(*                      ANALOG INPUT (%AI):     NONE                        *)
(*                     ANALOG OUTPUT (%AQ):     NONE                        *)
(*                                                                          *)
(****************************************************************************)








Program: DEMO30             D:\ACME\CONVEYOR\DEMO30                Block: _MAIN
```



```
01-15-93  16:23 GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v4.00)      Page    3

 |[  START OF LD  PROGRAM DEMO30    ]       (*                           *)
 |[     VARIABLE DECLARATIONS       ]
 |[       BLOCK DECLARATIONS        ]

                +--------+
        SUBR  1 |LIGHTS  |  LANG: LD        (* Shift lights controlled by timer *)
                +--------+

                +--------+
        SUBR  2 |DISPLAY|   LANG: LD        (* Move data from %R2 to lamps      *)
                +--------+

                +--------+
        SUBR  3 |  LOG   |  LANG: LD        (* Count shifts of the lights       *)
                +--------+

                +--------+
        SUBR  4 |MANUAL  |  LANG: LD        (* Manual setup and control         *)
                +--------+

                +--------+
        SUBR  5 |GENIUS  |  LANG: LD        (* Control genius demo case         *)
                +--------+

     #0001  SUBR 01
     #0002  SUBR 02
     #0003  SUBR 03
     #0004  SUBR 04
     #0005  SUBR 05

 |[     START OF PROGRAM LOGIC      ]
 |
 |(*  COMMENT  *)
 |   #0001  01    NOOP
 |
 | << RUNG 5  STEP #0002 >>
 |
 | LS1                                                                      AUTO
+--] [-----------------------------------------------------------------------(S)--
 |
 |   #0002  LD        %I0001
 |   #0003  SET       %Q0001
 |

 Program: DEMO30            D:\ACME\CONVEYOR\DEMO30                 Block: _MAIN
```

```
01-15-93  16:23 GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v4.00)      Page   4

|  << RUNG 6   STEP #0004 >>
|
|  METAL                                                                  AUTO
+--] [--+---------------------------------------------------------------(R)--
|       |
|FST_SCN|
+--] [--+
|
|       #0004  LD          %I0002
|       #0005  OR          %S0001
|       #0006  RST         %Q0001
|
|  << RUNG 7   STEP #0007 >>
|
|  AUTO    +-------------+ +-------------+
+--] [---+ CALL LIGHTS +-+ CALL    LOG  +
|         | (SUBROUTINE)| | (SUBROUTINE)|
|         +-------------+ +-------------+
|
|       #0007  LD          %Q0001
|       #0008  FUNC 90     CALLSUB
|                   P1:     00001
|       #0009  FUNC 90     CALLSUB
|                   P1:     00003
|
|  << RUNG 8   STEP #0010 >>
|
|  AUTO    +-------------+
+--]/[---+ CALL MANUAL +
|         | (SUBROUTINE)|
|         +-------------+
|
|       #0010  LD    NOT   %Q0001
|       #0011  FUNC 90     CALLSUB
|                   P1:     00004
|
|  << RUNG 9   STEP #0012 >>
|
|  AUTO    +-------------+
+--] [---+ CALL GENIUS +
|         | (SUBROUTINE)|
|         +-------------+
|
|       #0012  LD          %Q0001
|       #0013  FUNC 90     CALLSUB
|                   P1:     00005
|
| [      END OF PROGRAM LOGIC       ]
|
|       #0014  END OF PROGRAM
|


 Program: DEMO30              D:\ACME\CONVEYOR\DEMO30              Block: _MAIN
```

```
01-15-93  16:24 GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v4.00)      Page    1



                  GGGG EEEEE        FFFFF  AAA  N   N U   U  CCCC
                 G     E            F     A   A NN  N U   U C
                 G GGG EEEE         FFF   AAAAA N N N U   U C
                 G   G E            F     A   A N  NN U   U C
                  GGG  EEEEE        F     A   A N   N  UUU   CCCC



            AAA  U   U TTTTT  OOO  M   M  AAA  TTTTT IIIII  OOO  N   N
           A   A U   U   T   O   O MM MM A   A   T     I   O   O NN  N
           AAAAA U   U   T   O   O M M M AAAAA   T     I   O   O N N N
           A   A U   U   T   O   O M   M A   A   T     I   O   O N  NN
           A   A  UUU    T    OOO  M   M A   A   T   IIIII  OOO  N   N


(********************************************************************************)
(*                                                                              *)
(*                               Program:  DEMO30                               *)
(*                                                                              *)
(*        PLC PROGRAM ENVIRONMENT                HIGHEST REFERENCE USED         *)
(*       ---------------------------           ---------------------------      *)
(*                 INPUT (%I):   512                      INPUT:  %I0200        *)
(*                OUTPUT (%Q):   512                     OUTPUT:  %Q0010        *)
(*              INTERNAL (%M):  1024                   INTERNAL:  %M0001        *)
(*            GLOBAL DATA (%G):  1280                GLOBAL DATA:      NONE      *)
(*             TEMPORARY (%T):   256                  TEMPORARY:      NONE      *)
(*              REGISTER (%R):  2048                   REGISTER:  %R0002        *)
(*          ANALOG INPUT (%AI):   128               ANALOG INPUT:      NONE      *)
(*         ANALOG OUTPUT (%AQ):    64              ANALOG OUTPUT:      NONE      *)
(*                                                                              *)
(*                      PROGRAM SIZE (BYTES):       848                         *)
(*                                                                              *)
(*                                                                              *)
(********************************************************************************)











Program: DEMO30             D:\ACME\CONVEYOR\DEMO30
```

```
01-15-93  16:24 GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v4.00)      Page     2















(*****************************************************************************)
(*                                                                           *)
(*                              BLOCK:  LIGHTS                               *)
(*                                                                           *)
(*                                                                           *)
(*                           BLOCK SIZE (BYTES):   151                       *)
(*                      DECLARATIONS (ENTRIES):     2                        *)
(*                                                                           *)
(*                                                                           *)
(*                        HIGHEST REFERENCE USED                             *)
(*                     -----------------------------                         *)
(*                                                                           *)
(*                                 INPUT (%I):  %I0200                       *)
(*                                OUTPUT (%Q):  %Q0010                       *)
(*                              INTERNAL (%M):  %M0001                       *)
(*                           GLOBAL DATA (%G):     NONE                      *)
(*                             TEMPORARY (%T):     NONE                      *)
(*                              REGISTER (%R):  %R0002                       *)
(*                         ANALOG INPUT (%AI):     NONE                      *)
(*                        ANALOG OUTPUT (%AQ):     NONE                      *)
(*                                                                           *)
(*****************************************************************************)











Program: DEMO30            D:\ACME\CONVEYOR\DEMO30      Block: LIGHTS  (SUBR 01)
```

```
01-15-93  16:24 GE FANUC SERIES 90-30/90-20 DOCUMENTATION (v4.00)      Page    3

|[  START LD SUBROUTINE  LIGHTS   ]
|[     VARIABLE DECLARATIONS      ]
|[   START OF SUBROUTINE LOGIC    ]
|(*  COMMENT  *)
|
|  << RUNG 4  STEP #0002 >>
|
|%M0001  SFT_RT  +-----+
+--] [-----]7[---+ SHL |
|                | WORD|
|                |     |
|        %R0001 -+IN B2+-
|                | LEN |
|                |00001|
|         CONST -+N   Q+-%R0002
|         00001  |     |
|ALW_ON          |     |
+--]7[-----------+B1   |
|                +-----+
|
| << RUNG 5  STEP #0006 >>
|
| LASTBIT                                                                 SFT_RT
+--] [-------------------------------------------------------------------(SM)-
|
|[    END OF SUBROUTINE LOGIC     ]
|

Program: DEMO30              D:\ACME\CONVEYOR\DEMO30      Block: LIGHTS  (SUBR 01)
```

Chapter 10 I/O Configuration

The I/O configuration function is used to specify the mapping of logical references used by the user program to the physical I/O modules. This chapter contains the following sections:

Section 1: Default Configuration

When the PLC is first powered up or configuration is cleared, a default I/O configuration is created. The default I/O configuration is based on the I/O modules installed in the system. If you are satisfied with the default configuration, no further configuration is required.

The High Speed Counter, Axis Positioning, and Genius Communications modules are not included in the default I/O configuration; they must be manually configured using the Logicmaster 90 I/O configuration function.

For those users who want their system configured differently from the default (additional I/O modules, different I/O references, etc.), system configuration can be changed using the Logicmaster 90 I/O configuration function, described in the following sections of this chapter. If you edit the default I/O configuration or create a new one and store it to the PLC, it will replace the default configuration; and the PLC will not automatically configure itself again until the configuration is cleared.

The PLC will automatically configure the system based on the I/O modules installed according to the following table. This table shows how I/O references are assigned to each slot in the PLC. The 5-slot Model 311 or 313 PLC will have I/O addresses assigned to every slot. The 10-slot Model 311 or 313 PLC will have discrete I/O addresses assigned to each slot, but slots 9 and 10 will not be assigned analog I/O addresses. The Model 331 and higher PLCs will have analog and discrete addresses assigned to 15 of its slots (rack 0, slot 2 to rack 1, slot 6).

Table 10-1. Default I/O Configuration

Rack	Slot	Discrete Input	Discrete Output	Analog Input	Analog Output	Notes
0	1	%I001-032	%Q001-032	%AI001-008	%AQ001-004	This slot not configured in Model 331 or 341.
0	2	%I033-064	%Q033-064	%AI009-016	%AQ005-008	
0	3	%I065-096	%Q065-096	%AI017-024	%AQ009-012	
0	4	%I097-128	%Q097-128	%AI025-032	%AQ013-016	
0	5	%I129-160	%Q129-160	%AI033-040	%AQ017-020	Last slot in 5-slot Model 311 or 313.
0	6	%I161-192	%Q161-192	%AI041-048	%AQ021-024	
0	7	%I193-224	%Q193-224	%AI049-056	%AQ025-028	
0	8	%I225-256	%Q225-256	%AI057-064	%AQ029-032	Last slot to receive analog configuration in the 10-slot Model 311 or 313.
0	9	%I257-288	%Q257-288	%AI065-072	%AQ033-036	
0	10	%I289-320	%Q289-320	%AI073-080	%AQ037-040	Last slot in 10-slot Model 311 or 313.
1	1	%I321-352	%Q321-352	%AI081-088	%AQ041-044	
1	2	%I353-384	%Q353-384	%AI089-096	%AQ045-048	
1	3	%I385-416	%Q385-416	%AI097-104	%AQ049-052	
1	4	%I417-448	%Q417-448	%AI105-112	%AQ053-056	
1	5	%I449-480	%Q449-480	%AI113-120	%AQ057-060	
1	6	%I481-512	%Q481-512	%AI121-128	%AQ061-064	Last slot in Model 331 or higher.
1	7	–	–	–	–	
1	8	–	–	–	–	
1	9	–	–	–	–	
1	10	–	–	–	–	

You may view the current configuration by loading the configuration from the PLC to the programmer. (Refer to chapter 8, "Program Utilities," for more information about loading configuration from the PLC to the programmer.) The configuration can then be viewed from the I/O configuration function and can be printed using the configuration print function.

The PLC remains in default configuration mode until a configuration that is entered or modified by the Logicmaster 90-30/20/Micro configuration software is stored to the PLC. Only the CPU communication parameters (baud rate, parity, etc.) may be changed without taking the PLC out of default configuration mode. To put the PLC back into default configuration mode so that it will automatically configure the system, follow these steps:

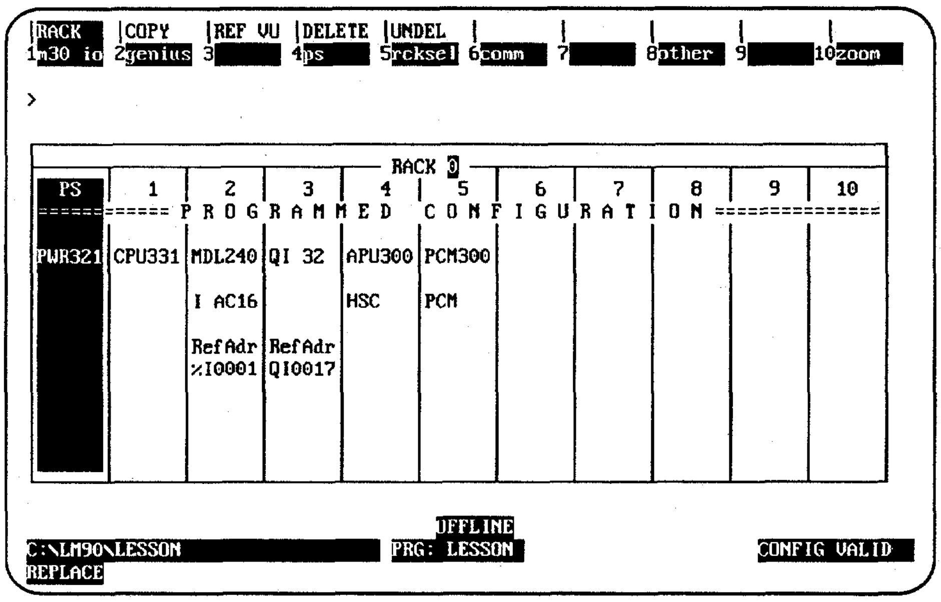
1. Press **Clear** (**`F5`**) from the Program Utility Functions menu to clear the configuration.
2. Verify that the selection for configuration is set to **`Y`** (Yes).
3. If you do not want to clear the ladder logic program or reference tables for that program, enter **`N`** (No) for program logic or reference tables.
4. Press the **`Enter`** key to begin the clear function. *Once the clear operation has begun, it cannot be aborted.* When the operation is complete, the software displays the message "Clear complete."

The PLC will now automatically configure the system using the default I/O configuration shown in table 10-1.

5. Press the **Load** function key (**`F1`**) to load the configuration.

Section 2: Displaying the I/O Configuration Rack Screen

When I/O (F1) is selected from the Configuration Software main menu, the I/O Configuration Rack screen, similar to the one shown below, is displayed. The following example represents a sample Series 90-30 PLC Model 331 system.

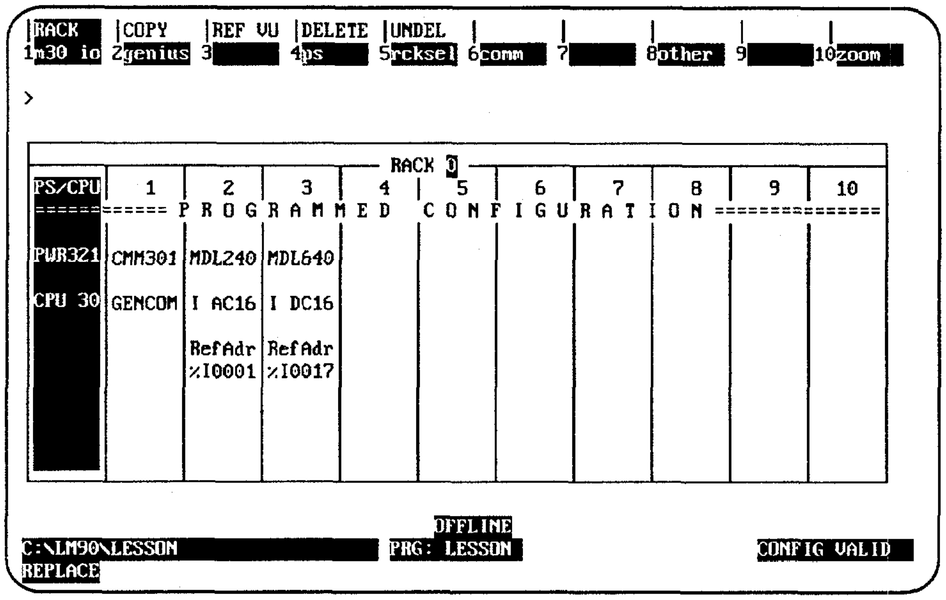


```
|RACK    |COPY    |REF VU |DELETE |UNDEL   |        |        |        |        |
1m30 io  2genius  3       4ps     5rcksel  6comm    7        8other   9        10zoom

>

                                   RACK 0
  PS   |  1    |  2    |  3    |  4    |  5    |  6    |  7    |  8    |  9    |  10
=============== P R O G R A M M E D   C O N F I G U R A T I O N ===============

PWR321 |CPU331 |MDL240 |QI 32  |APU300 |PCM300 |

       |       |I AC16 |       |HSC    |PCM    |

       |       |RefAdr |RefAdr |
       |       |%I0001 |Q10017 |

                                             OFFLINE
C:\LM90\LESSON                               PRG: LESSON                 CONFIG VALID
REPLACE
```

The rack shown above is divided into 11 slots. The first slot on the left is labeled "PS" at the top and always contains the power supply. The remaining slots are numbered 1 through 10. The CPU module always occupies slot 1 of rack 0. The remaining slots may contain any Series 90-30 I/O module, Genius Communications Module, Axis Positioning Module, or Programmable Coprocessor Module.

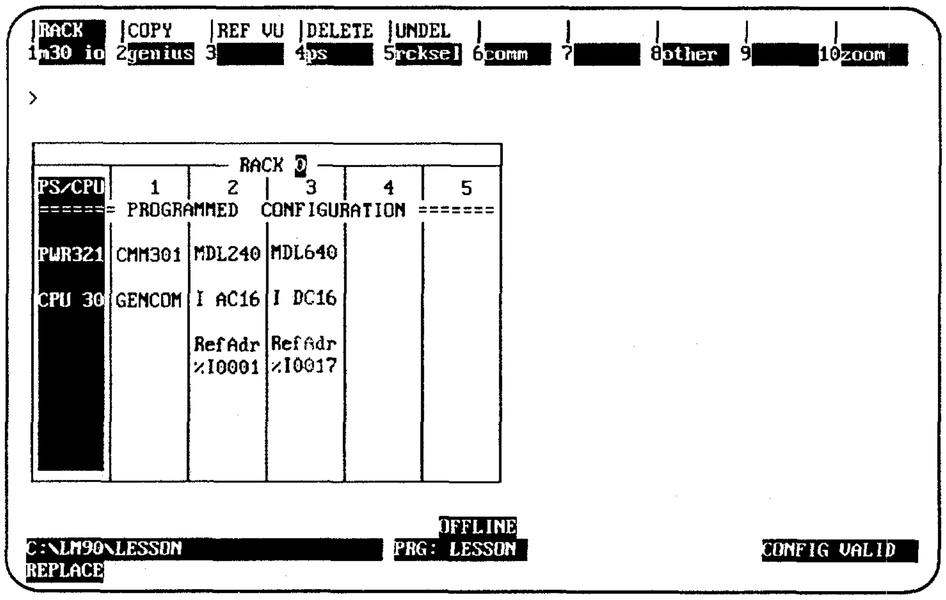
Model 331 and higher CPUs can have expansion racks. Up to four subsystem racks are allowed (up to seven subsystem racks in a 351 CPU). The main CPU rack is rack 0, and expansion racks are numbered 1 through 7. (All five racks default to be 10-slot racks.) The size of each rack can be changed on an individual rack basis by pressing **Rack Selection (F5)** and selecting the desired rack. This allows a mixture of 5-slot and 10-slot racks in a Series 90-30 PLC configuration. The **Up/Down** cursor movement keys and **Page Up/Down** keys are used to move between racks in ascending or descending order, respectively.

Both the 5-slot and 10-slot racks of the Model 311 PLC and the Model 313 PLC are one-rack systems. Moving from rack to rack is not allowed. The following example represents a sample 10-slot rack of the Model 311 PLC. Notice that on Models 311, 321, 323, and 313, Slot 1 *can* be used for I/O.



```
|RACK   |COPY    |REF VU  |DELETE  |UNDEL   |        |        |        |        |
1m30 io 2genius  3        4ps      5rcksel  6comm    7        8other   9        10zoom

>

                                  RACK 0
PS/CPU|   1   |   2   |   3   |   4   |   5   |   6   |   7   |   8   |   9   |  10
============ P R O G R A M M E D   C O N F I G U R A T I O N ================

PWR321| CMM301| MDL240| MDL640|
CPU 30| GENCOM| I AC16| I DC16|
      |       | RefAdr| RefAdr|
      |       | %I0001| %I0017|

                                             OFFLINE
C:\LM90\LESSON                               PRG: LESSON                  CONFIG VALID
REPLACE
```

The following example represents a sample 5-slot rack of the Model 311 PLC.



```
|RACK   |COPY    |REF VU  |DELETE  |UNDEL   |        |        |        |        |
1m30 io 2genius  3        4ps      5rcksel  6comm    7        8otherS  9        10zoom

>

                     RACK 0
PS/CPU|   1   |   2   |   3   |   4   |   5
======= PROGRAMMED  CONFIGURATION =======

PWR321| CMM301| MDL240| MDL640|
CPU 30| GENCOM| I AC16| I DC16|
      |       | RefAdr| RefAdr|
      |       | %I0001| %I0017|

                                             OFFLINE
C:\LM90\LESSON                               PRG: LESSON                  CONFIG VALID
REPLACE
```

The cursor position on the I/O Configuration Rack screen is indicated by having the slot highlighted in reverse video. Upon entering the Rack screen, the Power Supply slot is shown in reverse video. Use the Left or Right cursor keys to move the cursor from one slot to the next. To display another rack, use the Next and Previous page keys or the Up and Down cursor keys.

The rack screen presents an overview perspective of the Series 90-30 PLC system.

Function Key	Function	Description
F1	M30 I/O	Add Model 30 I/O modules.
F2	Genius	Add a Genius Communications Module.
F4	Power Supply	Add the power supply. If the power supply already exists, press **Zoom** (**F10**) to configure it.
F5	Rack Selection	Change the rack selection.
F6	Communications	Add a Communications Module.
F8	Other	Add other modules (e.g., PCM, ADC).
F10	Zoom	Display the current configuration of a slot.

Configuration Validation

CONFIG VALID is displayed in the lower right corner of each display screen after the configuration is successfully validated. When CONFIG INVALID is indicated, the file may not be stored to the PLC.

The most common cause of the CONFIG INVALID status is fatal overlaps among %I or %AI references.

A warning occurs when non-fatal overlaps occur; however, the configuration is still valid. Non-fatal overlaps are most likely to occur between references other than those described for fatal overlaps.

Copying Configuration from Slot to Slot

To copy configuration from slot to slot:

1. Position the cursor on the first module and press **Zoom** (**F10**) to display the slot configuration screen.
2. Configure the module.
3. Press **Rack** (**Shift-F1**) or the **Escape** key to return to the rack screen shown above.
4. With the cursor on that slot, press **Copy** (**Shift-F2**).
5. Move the cursor to another slot and press the **Enter** key. You can repeat this as many times as needed. As each module is entered, its reference address is set to the next highest available address.

   If the limit for the reference address has been reached, the copy function will adjust the module's reference address to the maximum allowed. Each copy after the maximum reference address has been reached will result in an address overlap.
6. When finished copying, exit copy mode by pressing the **Escape** key. The software displays the message "Copy Mode Ended".
7. For each copy, press **Zoom** (**F10**) to display the detail screen. Edit the reference address (if required) and other characteristics as needed.

Changing the Configuration of a Slot

Replacing the Module with Another of the Same Type

To change the configuration of a slot by replacing the module with another of the *same* type:

1. Place the cursor at the slot to be changed.
2. Press **Zoom** (`F10`) to display the module's detail screen.
3. Use one of the function keys to select a new module type for the slot. With the list of available modules displayed, move the cursor to the correct module and press the Enter key. Then, enter **Y** (Yes) after the prompt "REPLACE displayed module ? (Y/N)".
4. Complete the configuration; then press **Rack** (`Shift-F1`) or the **Escape** key to return to the rack display.

Replacing the Module with One of a Different Type

To change the configuration of a slot by replacing the module with one of a *different* type:

1. Place the cursor at the slot to be changed.
2. Press the function key that represents the type of module you want to place in the slot.
3. After confirming the deletion of the existing module, the module selection screen is displayed.
4. Complete the configuration; then, press **Rack** (`Shift-F1`) or the **Escape** key to return to the rack display.

Deleting the Configuration of a Module

To delete the configuration of a module:

1. Place the cursor on the module and press **Delete** (`Shift-F4`). Enter **Y** (Yes) after prompt "DELETE displayed module from slot ? (Y/N)". The configuration of the slot will be deleted.
2. To return the configuration to the slot, press **Undelete** (`Shift-F5`).

Moving a Module to Another Slot

To move the configuration of a module to another slot:

1. With the cursor at the configuration to be moved, press **Delete** (**`Shift-F4`**). Enter **`Y`** (Yes) after prompt "DELETE displayed module from slot ? (Y/N)". The configuration of the slot will be deleted.
2. Move the cursor to the new location; it may be in another rack.
3. Press **Undelete** (**`Shift-F5`**). The deleted configuration will appear in the new location. The reference address will not be affected.

Saving the Configuration to Disk

The configuration is automatically saved to a file on disk during the configuration process:

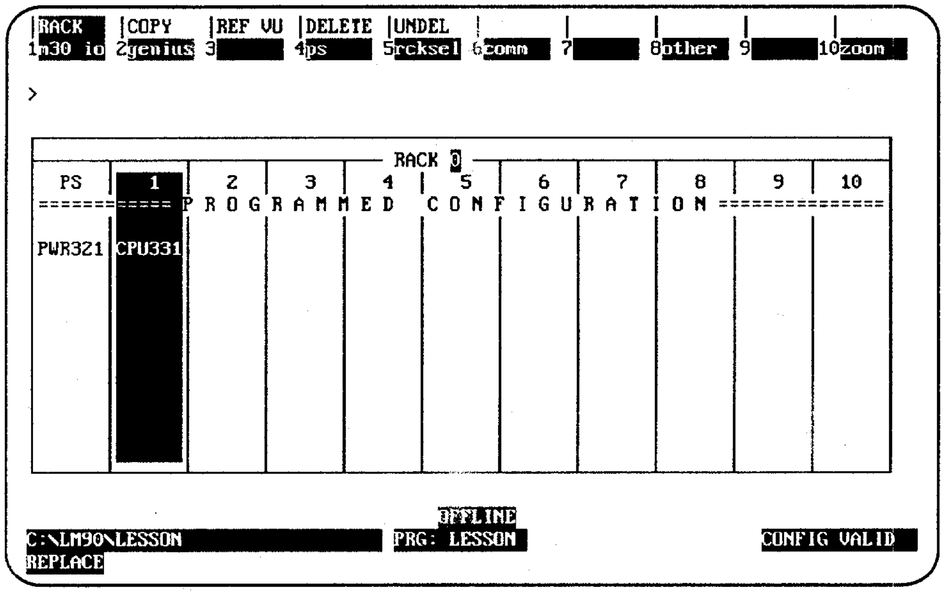
- When module data has been changed within a rack, and that new data is moved from that rack to another rack or when you cursor to a new rack.
- When module data has been changed within a rack, and then the **Escape** key is pressed to return to the Configuration Software main menu.
- When module data has been changed and **Reference View** (**`Shift-F3`**) is pressed.

The configuration may also be written to disk at any time by pressing **`ALT-U`** to update the disk.

Section 3: Configuring the CPU

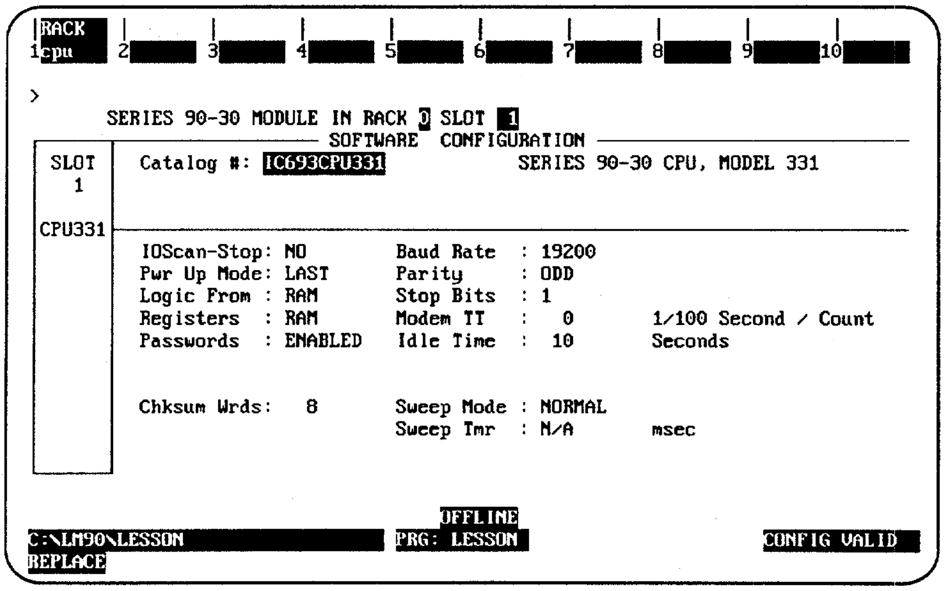
To configure the CPU module:

1. Model 331 or higher CPUs must be located in slot 1 of rack 0. Move the cursor to this slot by pressing the cursor movement keys. (Note that the Model 331 CPU is used in the following screens.)



```
|RACK   |COPY    |REF VU |DELETE |UNDEL  |       |       |       |       |
1m30 io 2genius  3       4ps     5rcksel 6comm   7       8other  9      10zoom
>
                                   RACK 0
  PS   |  1   |  2   |  3   |  4   |  5   |  6   |  7   |  8   |  9   |  10
============= P R O G R A M M E D   C O N F I G U R A T I O N ==============
PWR321 CPU331

                                       OFFLINE
C:\LM90\LESSON                         PRG: LESSON                 CONFIG VALID
REPLACE
```

2. Press **Zoom** (`F10`) to display the CPU detail screen:



```
|RACK |     |     |     |     |     |     |     |     |
1cpu  2     3     4     5     6     7     8     9    10
>
      SERIES 90-30 MODULE IN RACK 0 SLOT 1
                     SOFTWARE  CONFIGURATION
 SLOT   Catalog #: IC693CPU331          SERIES 90-30 CPU, MODEL 331
  1

CPU331
        IOScan-Stop: NO         Baud Rate  : 19200
        Pwr Up Mode: LAST       Parity     : ODD
        Logic From : RAM        Stop Bits  : 1
        Registers  : RAM        Modem TT   :   0      1/100 Second / Count
        Passwords  : ENABLED    Idle Time  :  10      Seconds


        Chksum Wrds:    8       Sweep Mode : NORMAL
                                Sweep Tmr  : N/A      msec

                                       OFFLINE
C:\LM90\LESSON                         PRG: LESSON                 CONFIG VALID
REPLACE
```

Parameter	Description
I/O Scan-Stop	Indicates whether the I/O is to be scanned while the PLC is in **STOP** mode. Choices are **YES** or **NO***.
Power-Up Mode	Indicates which state the PLC is to be powered up in. Choices are **RUN**, **STOP**, or **LAST***. **LAST** indicates that the PLC will power up in the same mode it powered down in.
Logic From	Indicates whether the program logic is stored in **RAM***, or in an additional, optional **EEPROM**.
Registers	Indicates whether the source of registers is **RAM*** or **EEPROM**.
Passwords	Indicates whether passwords are **ENABLED*** or **DISABLED**.
Baud Rate	Transmission rate in bits per second. Choices are **300**, **600**, **1200**, **2400**, **4800**, **9600**, or **19200***.
Parity	Type of parity bits added to each word. Choices are **ODD***, **EVEN**, or **NONE**.
Stop Bits	Communication uses at least one stop bit. Slower communication uses two stop bits. Choices are **1*** or **2**.
Modem Turnaround Time	Modem turnaround delay time counts. 1 count = 1/100 second. Choices are **0** to **255** (default = **0**).
Idle Time	Maximum communication idle time from **1** to **60** seconds (default = **10** seconds).
Checksum Words	The number of words of user program to be applied to the checksum function per sweep. Choices are **0** through **32**, inclusive.
Sweep Mode	Sweep mode can be either **NORMAL SWEEP** mode (**NORMAL***) or **CONSTANT SWEEP** mode (**CNST SWP**). To change the settings, tap the Down Arrow key till the cursor rests in the "Sweep Mode:" parameter. Then press the **Tab** key to change the selection from "NORMAL" to "CNST SWP" (if necessary—if you are just changing the time of the Constant Sweep, it may already say "CNST SWP"). Then tap the **Down Arrow** key once to move the cursor to the "Sweep Tmr" parameter and enter the number of milliseconds (from 5 to 200—5 to 500 for 351 CPUs) that you want. To store the changes to the folder, press the **Escape** key twice. Then store the configuration to the PLC and switch the PLC into RUN mode for this change to take effect. This Sweep Mode adjustment is different from the Active Constant Sweep which can be edited only in RUN mode. Refer to the "Standard Sweep Mode Variations" section of chapter 2 in the *Series 90-30/20/Micro Reference Manual* (GFK-0467) and page 5-19 and following of this manual for a discussion of these two types of Constant Sweep Mode. In **NORMAL SWEEP** mode, the PLC sweep executes as fast as possible. The overall PLC sweep time depends on the logic program plus the time required to compute the checksum. In **CONSTANT SWEEP** mode, the overall PLC sweep time is fixed.
Sweep Timer	A byte value which can be configured, in 1 millisecond increments, to be any value from **5** to **200** ms. —5 to 500 for 351 CPUs. Default = **100**. <u>Note:</u> The sweep timer is not configurable when the sweep mode is **NORMAL**. In this mode, the timer is set to N/A (Not Applicable), and you cannot cursor to the *Sweep Timer* field.

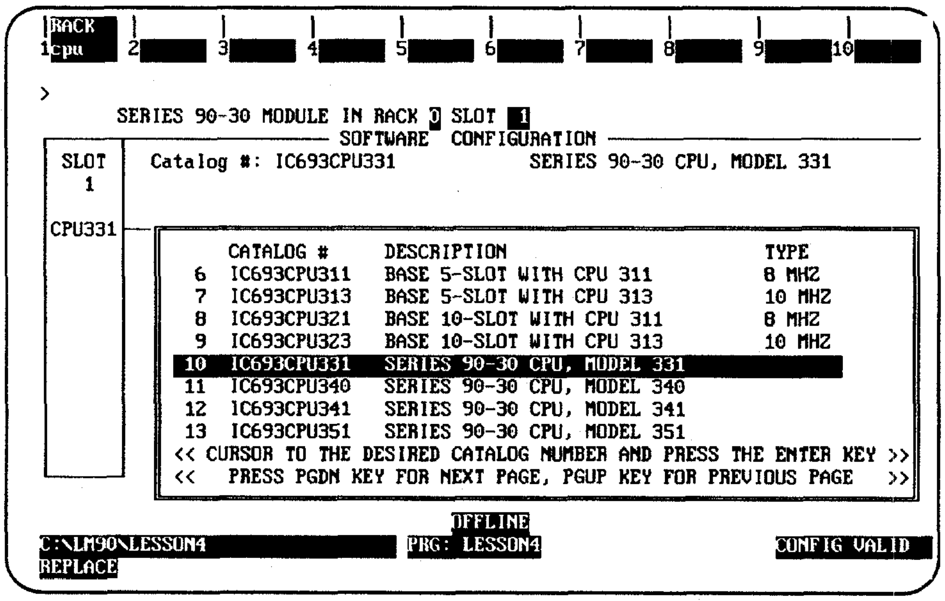
* Default selection.

Caution

In order to re-enable passwords once they have been disabled, PLC memory must be cleared with an HHP. The HHP needs to be connected. Then power off the PLC. Then hold both the <CLR> and <M/T> keys down while powering the PLC back up. If you do not have an HHP, call the GE Fanuc Technical Service Hotline (1-800-828-5747) for assistance.

3. Use the **Tab** key and **Back Tab** key (**Shift-Tab**) to scroll through the selections for each parameter displayed on this screen. For help selecting the parameters, press **ALT-H**.

4. When a value has been selected for each parameter, press **CPU** (**F1**) to display a list of catalog numbers and modules.



```
RACK
1cpu   2    3    4    5    6    7    8    9    10

>
       SERIES 90-30 MODULE IN RACK 0 SLOT 1
                        SOFTWARE  CONFIGURATION
 SLOT    Catalog #: IC693CPU331           SERIES 90-30 CPU, MODEL 331
  1

CPU331
              CATALOG #     DESCRIPTION                      TYPE
          6   IC693CPU311   BASE 5-SLOT WITH CPU 311         8 MHZ
          7   IC693CPU313   BASE 5-SLOT WITH CPU 313         10 MHZ
          8   IC693CPU321   BASE 10-SLOT WITH CPU 311        8 MHZ
          9   IC693CPU323   BASE 10-SLOT WITH CPU 313        10 MHZ
         10   IC693CPU331   SERIES 90-30 CPU, MODEL 331
         11   IC693CPU340   SERIES 90-30 CPU, MODEL 340
         12   IC693CPU341   SERIES 90-30 CPU, MODEL 341
         13   IC693CPU351   SERIES 90-30 CPU, MODEL 351
        << CURSOR TO THE DESIRED CATALOG NUMBER AND PRESS THE ENTER KEY >>
        <<   PRESS PGDN KEY FOR NEXT PAGE, PGUP KEY FOR PREVIOUS PAGE   >>

                                   OFFLINE
C:\LM90\LESSON4                  PRG: LESSON4              CONFIG VALID
REPLACE
```

5. Position the cursor on the catalog number for the Model 331 CPU (IC693CPU331), and press the **Enter** key.
6. Press **Rack** (**`Shift-F1`**) or the **Escape** key to return to the rack display.

Selecting a Different CPU Module

Note

If you configure a newly-released CPU using Release 4 or later of Logicmaster 90-30/20 software, you cannot use this configuration with an earlier release of the programming software. That is to say, if you configure a CPU with a recent release of the software, use the same release (or later) for programming.

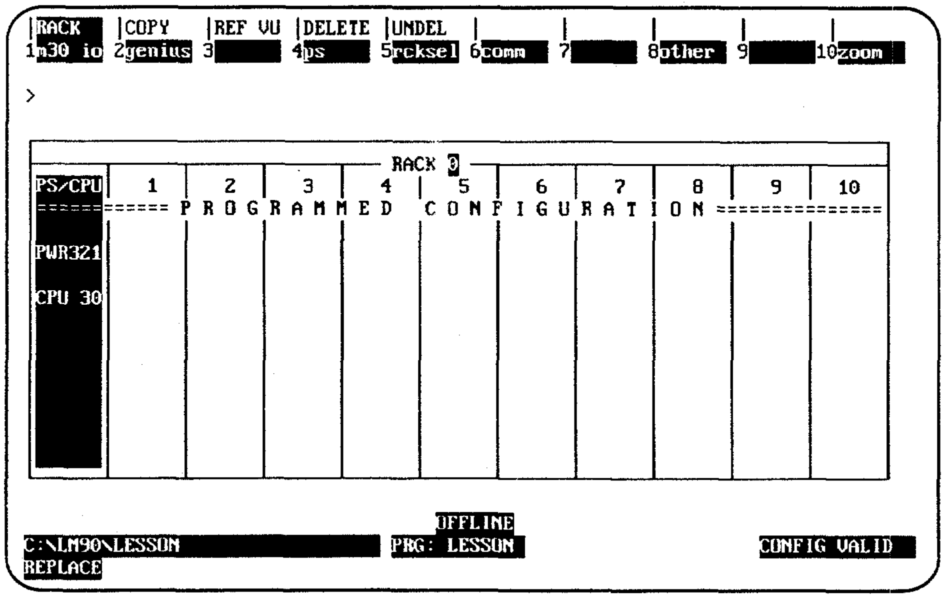
1. To select a different CPU module, move the cursor to the catalog number for the desired CPU type and press the **Enter** key. Then, enter **`Y`** (Yes) after the prompt "REPLACE displayed module ? (Y/N)".

   For example, to configure a 10-slot rack for the Model 311 CPU, move the cursor to the entry for that module (IC693CPU321) and press the **Enter** key.

2. If any modules (e.g., intelligent modules) that are not supported by the 10-slot rack have been configured, the following error message is displayed:

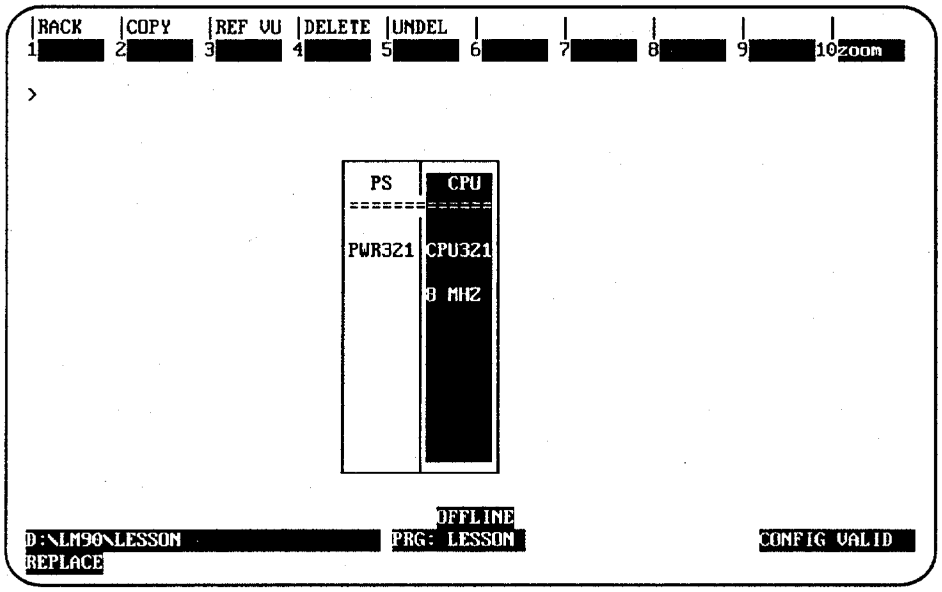
```
Selected CPU does not support all currently configured modules
```

3. Press the **Escape** key twice, once to zoom out of the list of catalog numbers and a second time to zoom out of the CPU detail screen back to rack 0.
4. Use the delete function to delete modules which are not supported. Then, press **Zoom** (**`F10`**) to return to the list of catalog numbers.
5. Position the cursor once more on the catalog number for the 10-slot rack for the Model 311 CPU (IC693CPU321), and press the **Enter** key. Then, press the **Escape** key twice to return to the Rack Configuration screen shown below.



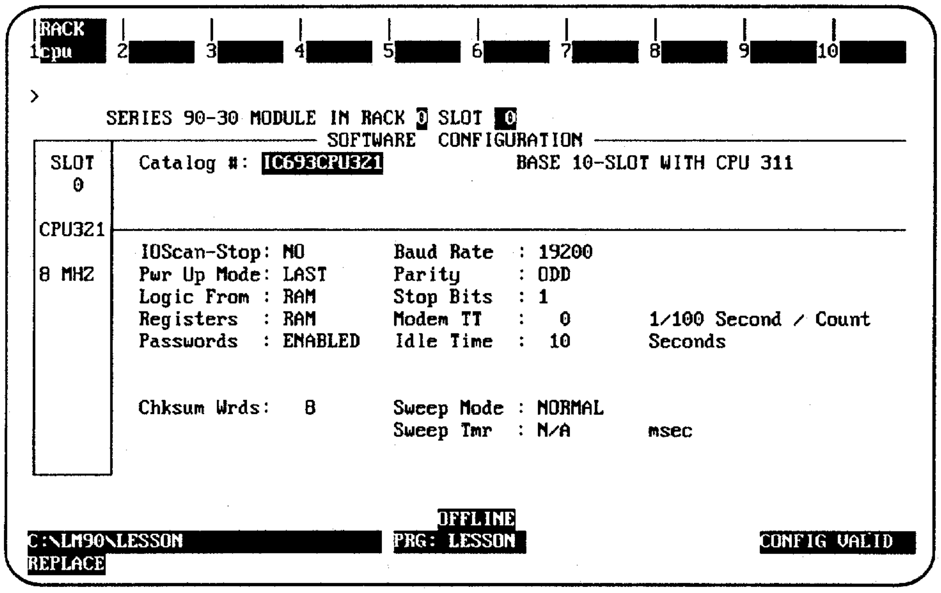
```
|RACK    |COPY    |REF VU |DELETE |UNDEL   |        |        |        |        |
1m30 io 2genius 3       4ps     5rcksel 6comm   7       8other  9       10zoom
>

                                  RACK 0
PS/CPU|   1   |   2   |   3   |   4   |   5   |   6   |   7   |   8   |   9   |  10
============ P R O G R A M M E D   C O N F I G U R A T I O N ==============
PWR321
CPU 30

                                       OFFLINE
C:\LM90\LESSON                         PRG: LESSON                 CONFIG VALID
REPLACE
```

6. Press **Zoom** (**F10**) to display the Power Supply/CPU screen.



```
|RACK    |COPY    |REF VU  |DELETE  |UNDEL   |        |        |        |        |
1        2        3        4        5        6        7        8        9        10zoom
>

                              PS    CPU
                             ============
                             PWR321 CPU321
                                    8 MHZ

                                           OFFLINE
D:\LM90\LESSON                             PRG: LESSON                CONFIG VALID
REPLACE
```

7. Then, press **Zoom** (**F10**) again to display the CPU detail screen.

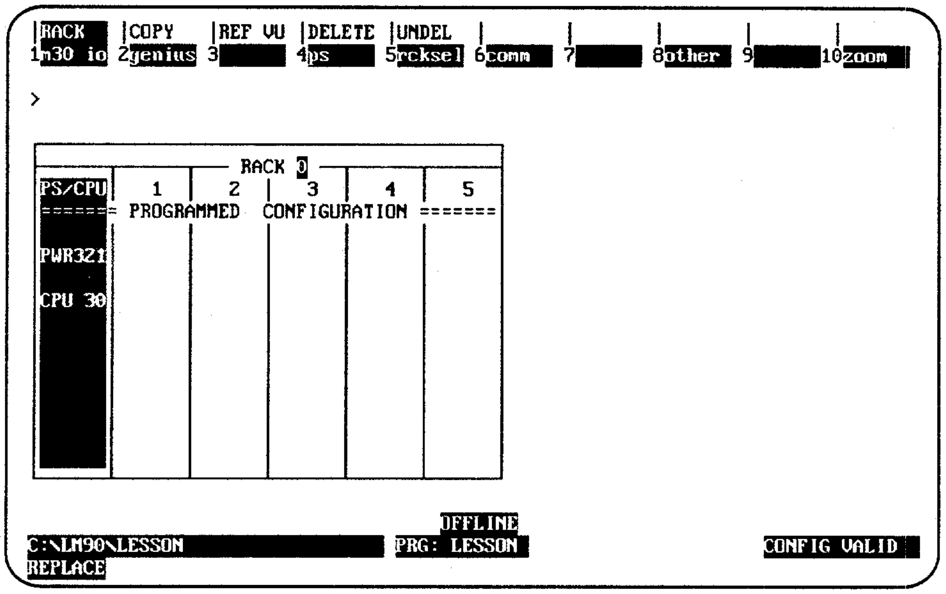


```
|RACK    |        |        |        |        |        |        |        |        |
1cpu     2        3        4        5        6        7        8        9        10
>
   SERIES 90-30 MODULE IN RACK 0 SLOT 0
                      SOFTWARE  CONFIGURATION
 SLOT   Catalog #: IC693CPU321          BASE 10-SLOT WITH CPU 311
  0

CPU321
        IOScan-Stop: NO         Baud Rate  : 19200
8 MHZ   Pwr Up Mode: LAST       Parity     : ODD
        Logic From : RAM        Stop Bits  : 1
        Registers  : RAM        Modem TT   :   0       1/100 Second / Count
        Passwords  : ENABLED    Idle Time  :  10       Seconds

        Chksum Wrds:    8       Sweep Mode : NORMAL
                                Sweep Tmr  : N/A       msec

                                           OFFLINE
C:\LM90\LESSON                             PRG: LESSON                CONFIG VALID
REPLACE
```

8. Press the **Escape** key once to return to the detail screen. Press the **Escape** key again to save the module configuration and return to the rack display.

9. Or, to configure the 5-slot rack for the Model 311 CPU, press **CPU** (**F1**) to display a list of catalog numbers and modules.

10. Move the cursor to the entry for the 5-slot rack (IC693CPU311), and press the **Enter** key. Then, enter **Y** (Yes) after the prompt "REPLACE displayed module ? (Y/N)".



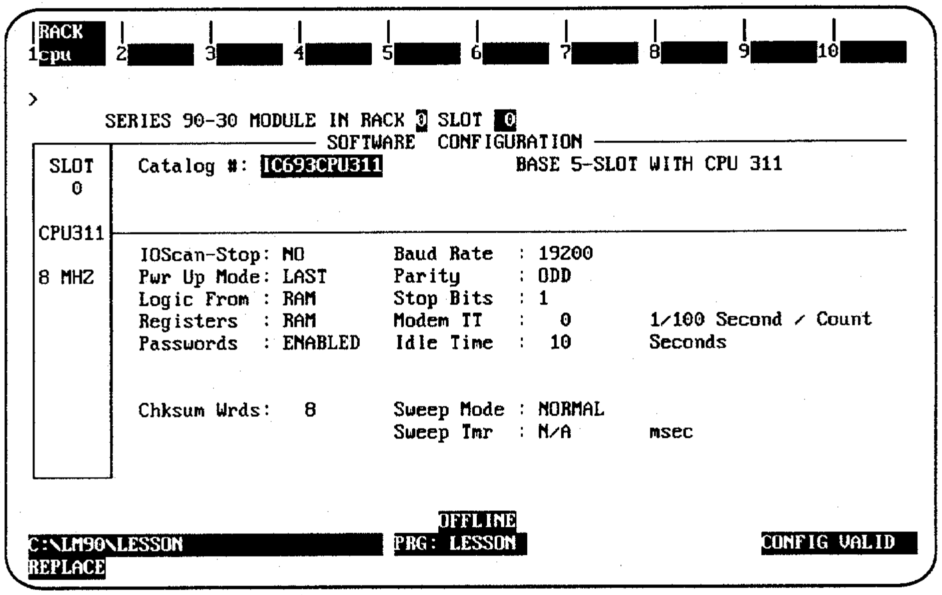
```
|RACK    |COPY    |REF VU |DELETE |UNDEL  |       |       |       |       |
1m30 io 2genius 3       4ps     5rcksel 6comm   7       8other  9       10zoom

>

                 RACK 0
PS/CPU|   1   |   2   |   3   |   4   |   5
======= PROGRAMMED  CONFIGURATION =======

PWR321

CPU 30

                                           OFFLINE
C:\LM90\LESSON                             PRG: LESSON               CONFIG VALID
REPLACE
```

Since no boards had been configured in slots 6 through 10, the change from a 10-slot rack to a 5-slot rack is allowed. However, if boards had been configured in any of slots 6 through 10, the following error message would be displayed:

```
Slots greater than 5 will be lost, Continue REPLACE?  (Y/N)
```

If **Y** (Yes) is entered, the system will automatically delete slots 6 through 10 before changing the CPU module.

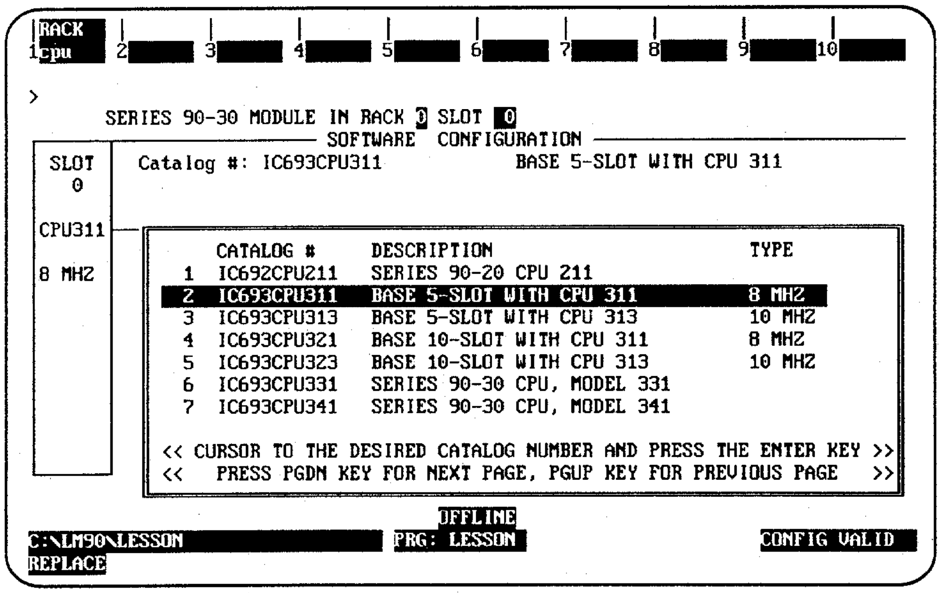
11. Press **Zoom** (**F10**) and then **CPU** (**F1**) to display the CPU detail screen for the 5-slot rack for the Model 311 CPU.



```
 RACK
1 cpu  2      3      4      5      6      7      8      9      10

>
      SERIES 90-30 MODULE IN RACK 0 SLOT 0
                           SOFTWARE  CONFIGURATION
 SLOT   Catalog #: IC693CPU311           BASE 5-SLOT WITH CPU 311
  0

 CPU311
        IOScan-Stop: NO        Baud Rate  : 19200
 8 MHZ  Pwr Up Mode: LAST      Parity     : ODD
        Logic From : RAM       Stop Bits  : 1
        Registers  : RAM       Modem TT   :  0      1/100 Second / Count
        Passwords  : ENABLED   Idle Time  :  10     Seconds

        Chksum Wrds:   8       Sweep Mode : NORMAL
                               Sweep Tmr  : N/A     msec

                                OFFLINE
C:\LM90\LESSON                  PRG: LESSON                 CONFIG VALID
REPLACE
```

Use the **Tab** key and **Back Tab** key (**Shift-Tab**) to select a value for each parameter displayed on this screen.

12. Then, press the **Escape** key twice to display the Rack Configuration screen.
13. Press **Zoom** (**F10**) twice and then press **CPU** (**F1**) to display the list of catalog numbers and modules.



```
 RACK
1 cpu  2      3      4      5      6      7      8      9      10

>
      SERIES 90-30 MODULE IN RACK 0 SLOT 0
                           SOFTWARE  CONFIGURATION
 SLOT   Catalog #: IC693CPU311           BASE 5-SLOT WITH CPU 311
  0

 CPU311
              CATALOG #     DESCRIPTION                        TYPE
 8 MHZ      1  IC692CPU211   SERIES 90-20 CPU 211
            2  IC693CPU311   BASE 5-SLOT WITH CPU 311           8 MHZ
            3  IC693CPU313   BASE 5-SLOT WITH CPU 313           10 MHZ
            4  IC693CPU321   BASE 10-SLOT WITH CPU 311          8 MHZ
            5  IC693CPU323   BASE 10-SLOT WITH CPU 313          10 MHZ
            6  IC693CPU331   SERIES 90-30 CPU, MODEL 331
            7  IC693CPU341   SERIES 90-30 CPU, MODEL 341

          << CURSOR TO THE DESIRED CATALOG NUMBER AND PRESS THE ENTER KEY >>
          <<   PRESS PGDN KEY FOR NEXT PAGE, PGUP KEY FOR PREVIOUS PAGE    >>

                                OFFLINE
C:\LM90\LESSON                  PRG: LESSON                 CONFIG VALID
REPLACE
```

14. Position the cursor on the catalog number for the Model 311 CPU (IC693CPU311), and press the **Enter** key.
15. Press the **Escape** key to return to the rack display.

Note

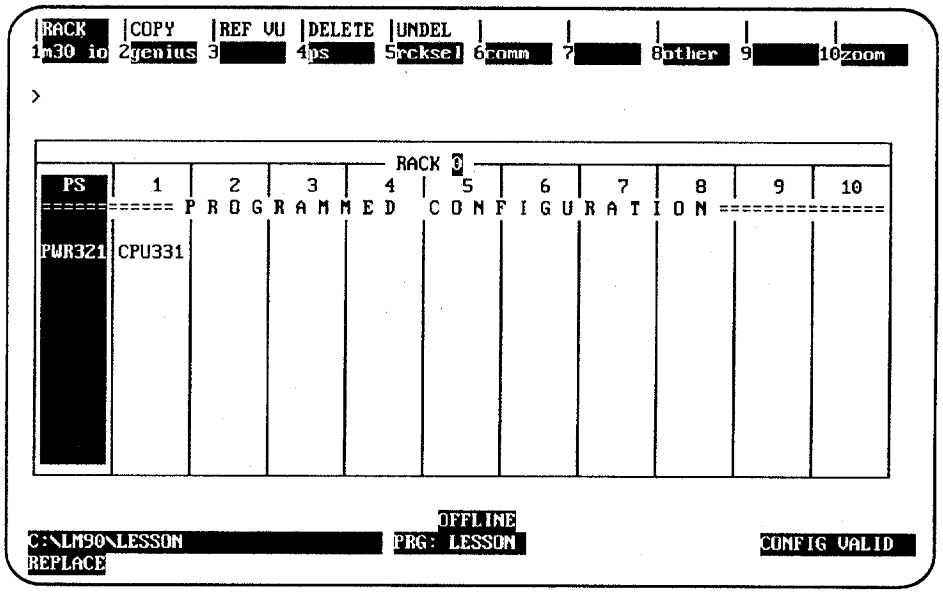
If you install and configure a 351 CPU and use folders you created previously, you will be prompted to convert the folder to one that uses the code written for 351 CPUs. Refer to page 7-4 for other considerations when using previously created folders with a 351 CPU.

Section 4: Selecting the Base Rack

The Model 331 CPU and Model 341 CPU are the only non-backplane-mounted CPUs for the Series 90-30 PLC. The Model 331 and 341 CPUs are plug-in modules, which can be removed from the baseplate.

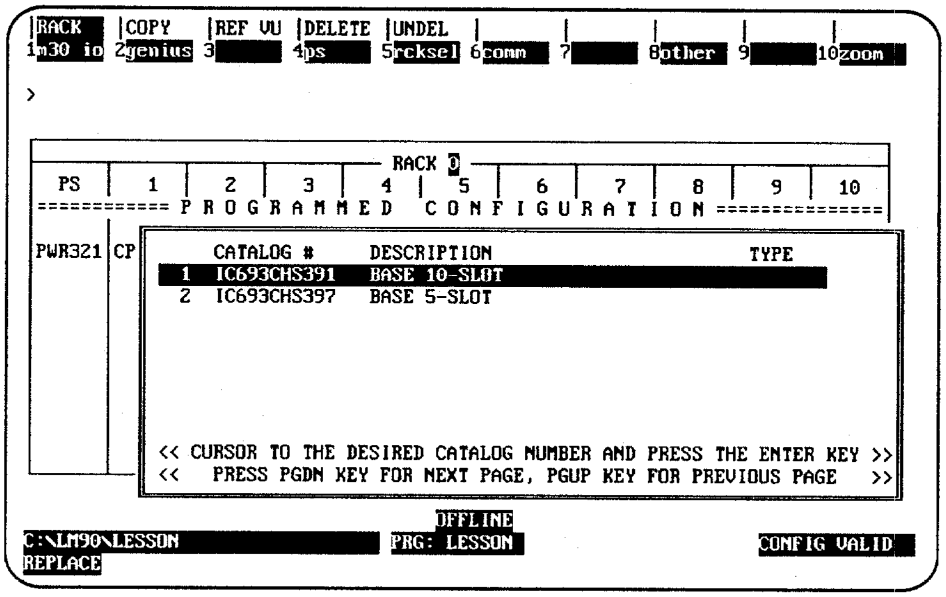
Both the Model 331 CPU and the Model 341 CPU can have expansion racks. Up to five subsystem racks are allowed. The main CPU rack is rack 0, and expansion racks are numbered 1 through 4. (All five racks default to be 10-slot racks.) The size of each rack can be changed on an individual rack basis, allowing a mixture of 5-slot and 10-slot racks in a Series 90-30 PLC configuration. The Up/Down cursor keys and Page Up/Down keys are used to move between racks in ascending or descending order, respectively.

1. Initially, a 10-slot rack is displayed, as shown below for the Model 331 PLC.



```
|RACK     |COPY     |REF VU  |DELETE  |UNDEL   |        |        |        |        |
1m30 io   2genius   3        4ps      5rcksel  6comm    7        8other   9        10zoom

>

                                   RACK 0
  PS   |  1   |  2   |  3   |  4   |  5   |  6   |  7   |  8   |  9   |  10
============ P R O G R A M M E D   C O N F I G U R A T I O N ===============
PWR321 |CPU331|

                                                      OFFLINE
C:\LM90\LESSON                               PRG: LESSON                 CONFIG VALID
REPLACE
```

2. When **Rack Selection (F5)** is pressed from any slot of rack 0 of a 5-slot 10-slot Model 331 CPU, a list of catalog numbers and racks is displayed:



```
|RACK    |COPY    |REF VU  |DELETE  |UNDEL   |        |        |        |        |
1m30 io 2genius  3       4ps      5rcksel  6comm    7        8other   9       10zoom
>

 ------------------------------ RACK 0 ------------------------------------
   PS    |   1   |   2   |   3   |   4  |  5   |   6   |   7   |   8   |   9   |  10
=========== P R O G R A M M E D   C O N F I G U R A T I O N ==============
PWR321 |CP|      CATALOG #       DESCRIPTION                      TYPE
          |   1  IC693CHS391     BASE 10-SLOT
          |   2  IC693CHS397     BASE 5-SLOT

          << CURSOR TO THE DESIRED CATALOG NUMBER AND PRESS THE ENTER KEY >>
          <<    PRESS PGDN KEY FOR NEXT PAGE, PGUP KEY FOR PREVIOUS PAGE   >>
                                       OFFLINE
C:\LM90\LESSON                         PRG: LESSON                  CONFIG VALID
REPLACE
```

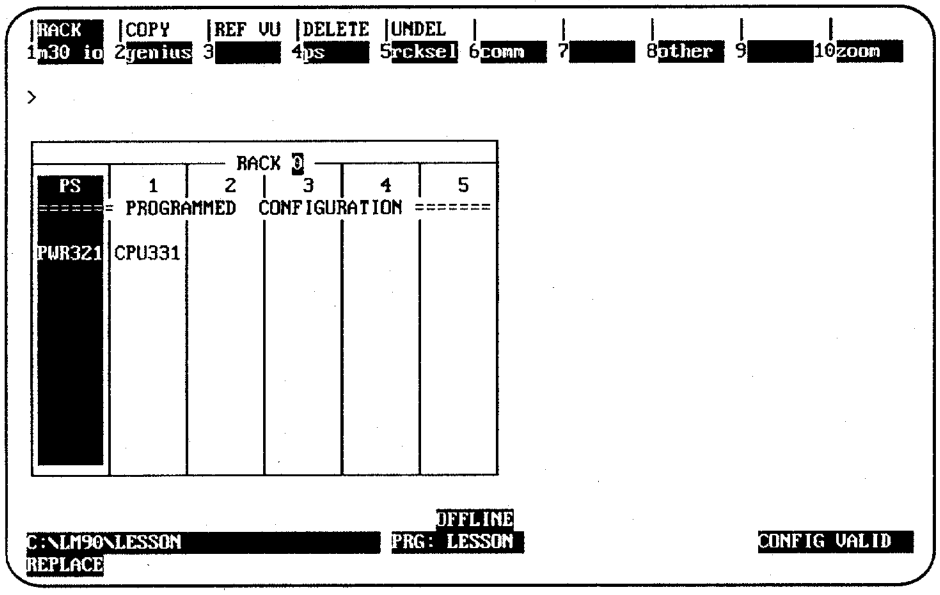
If the **Escape** key or **ALT-A** (Abort) is pressed while this list is displayed, the rack screen is redisplayed.

3. Position the cursor on the catalog number of the desired rack, and press the **Enter** key.

4. When switching from a 10-slot rack to a 5-slot rack, a validation check is performed to see if there are any configured modules in slots 6 through 10. If there are, the following message is displayed:

```
Slots greater than 5 will be lost, continue REPLACE ?  (Y/N)
```

If **N** (No) is entered, the current rack will remain displayed. If **Y** (Yes) is entered, the system will automatically delete slots 6 through 10 before changing the rack size. The 5-slot rack appears as shown below:



```
|RACK   |COPY   |REF VU |DELETE |UNDEL  |       |       |       |       |
1m30 io 2genius 3       4ps     5rcksel 6comm   7       8other  9       10zoom
>

                    RACK 0
  PS   |   1   |   2   |   3   |   4   |   5
======= PROGRAMMED  CONFIGURATION =======
PWR321  CPU331

                                   OFFLINE
C:\LM90\LESSON                     PRG: LESSON                 CONFIG VALID
REPLACE
```

When changing the rack selection, note these rules:

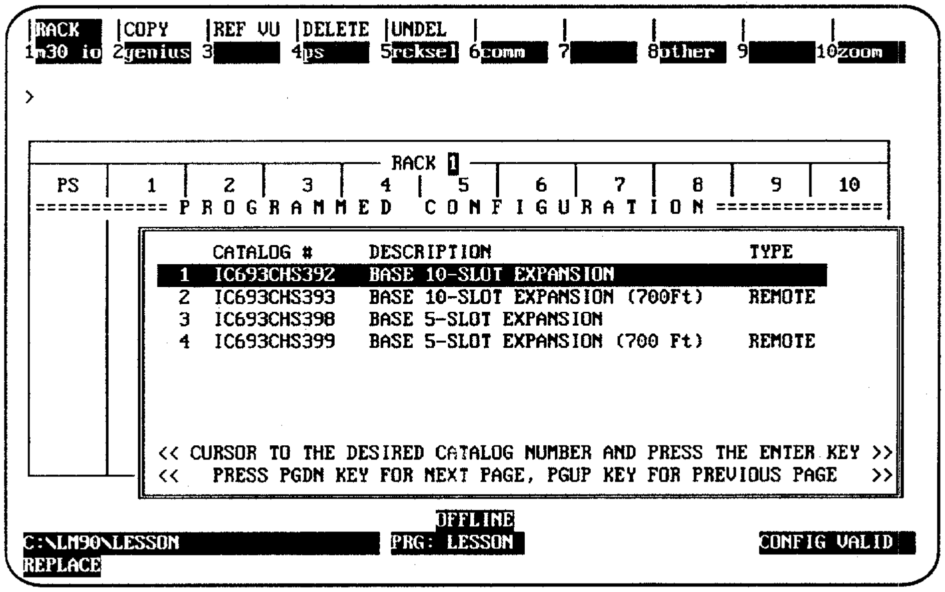
- The rack cursor will always be displayed on the Power Supply module after a successful rack change.
- Selecting the catalog number of the same size rack currently configured does not change the display.
- When a rack switch is made, the modules will have the same slot location they had in the previous rack.
- Rack 0, as well as any expansion rack, can be configured to be a 5-slot rack.
- When switching from a 5-slot rack to a 10-slot rack, no validation check is performed.
- When a previously configured 10-slot rack is changed to a 5-slot rack and then back to a 10-slot rack, the modules initially configured for the 10-slot rack are not restored.

Base Rack Power Supply

The base rack power supply is automatically configured when the Model 331 or 341 CPU is selected. The Power Supply (F4) function cannot be used to change the base rack (rack 0) power supply. You must first press **Zoom (F10)** to zoom into the power supply, and then press **Power Supply (F1)** to change the base rack (rack 0) power supply.

Selecting an Expansion Rack

Expansion rack catalog numbers are different from the base rack catalog numbers for a Model 311 or 341 CPU subsystem. When **Rack Selection (F5)** is pressed from any slot of rack 1 of a subsystem rack, a list of catalog numbers and expansion racks is displayed.



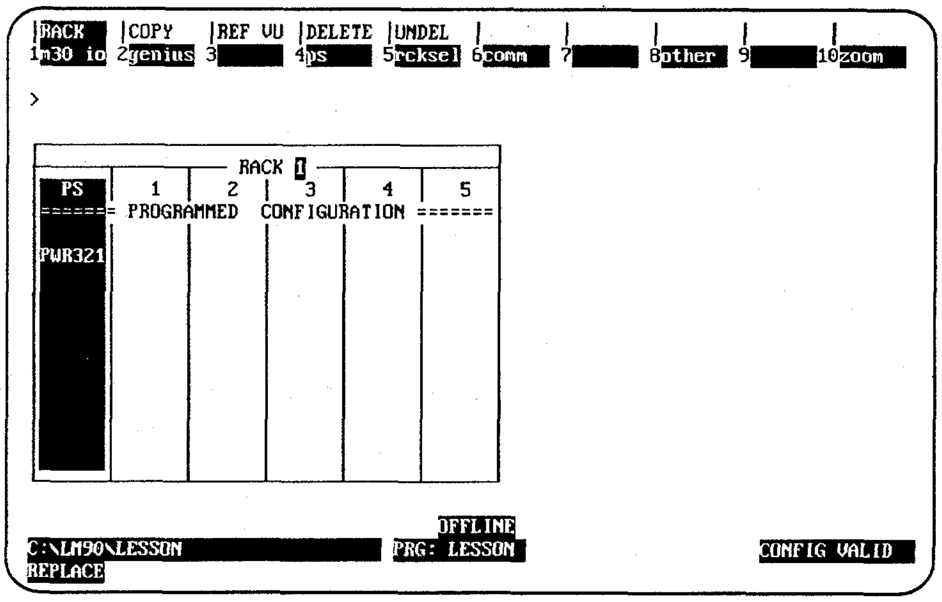
```
|RACK    |COPY    |REF VU  |DELETE  |UNDEL   |        |        |        |        |
1m30 io 2genius  3        4ps      5rcksel  6comm    7        8other   9        10zoom
>

                              RACK 1
  PS   |   1   |   2   |   3   |   4   |   5   |   6   |   7   |   8   |   9   |  10
============ P R O G R A M M E D   C O N F I G U R A T I O N ==============

              CATALOG #      DESCRIPTION                        TYPE
           1  IC693CHS392    BASE 10-SLOT EXPANSION
           2  IC693CHS393    BASE 10-SLOT EXPANSION (700Ft)     REMOTE
           3  IC693CHS398    BASE 5-SLOT EXPANSION
           4  IC693CHS399    BASE 5-SLOT EXPANSION (700 Ft)     REMOTE

         << CURSOR TO THE DESIRED CATALOG NUMBER AND PRESS THE ENTER KEY >>
         <<   PRESS PGDN KEY FOR NEXT PAGE, PGUP KEY FOR PREVIOUS PAGE   >>

                                         OFFLINE
C:\LM90\LESSON                           PRG: LESSON                CONFIG VALID
REPLACE
```

Note

The I/O bus of the base rack is extended to the I/O bus of expansion racks via an I/O bus expansion cable.

To select an expansion rack (for example, the 5-slot expansion rack), position the cursor on the catalog number for that rack (for this example, IC693CHS398), and press the **Enter** key.

The following rack is displayed if a 10-slot expansion rack with no modules is switched to a 5-slot expansion rack.



```
|RACK    |COPY    |REF VU  |DELETE  |UNDEL   |        |        |        |        |
1m30 io 2genius 3        4ps      5rcksel 6comm    7        8other   9        10zoom

>

                 RACK 1
  PS   |   1    |   2  |   3    |   4    |   5
====== = PROGRAMMED  CONFIGURATION =======

PWR321

                                          OFFLINE
C:\LM90\LESSON                          PRG: LESSON                   CONFIG VALID
REPLACE
```

Note

No power supply is configured until an I/O module is configured or the Power Supply (F4) key is used to select one.

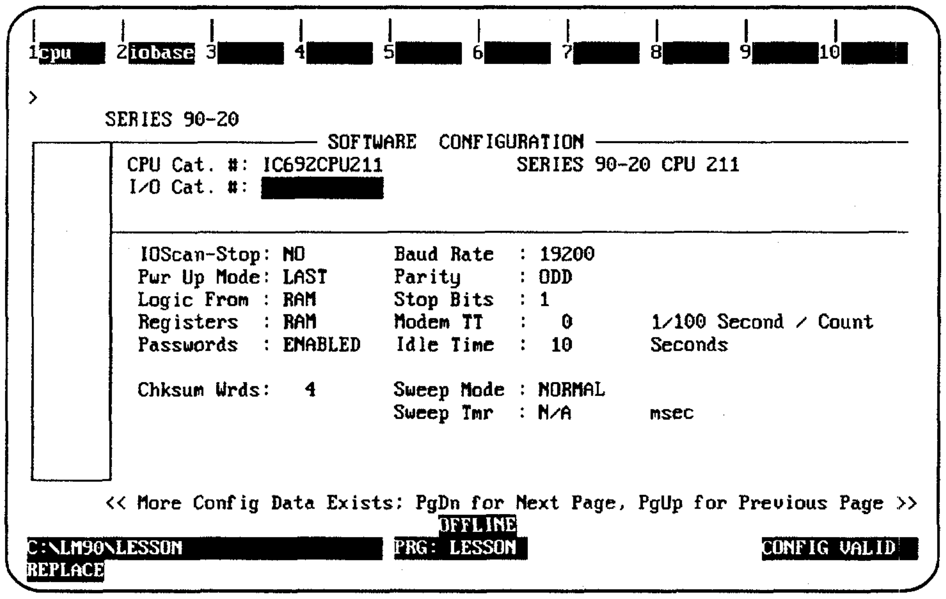
Section 5: Configuring the Model 211 CPU

The I/O configuration of the CPU 211 is similar to the configuration of a 5-slot CPU 311, which has been configured with a power supply, CPU, generic discrete input, generic discrete output, and High Speed Counter modules. The reference address assignments for the CPU 211 I/O are fixed.

No rack screen is displayed for the 211 PLC.

1. Press **CPU** (**F1**) to display a list of catalog numbers and modules. Cursor to the desired CPU catalog number, and press the **Enter** key to select a new CPU.

   If a Series 90-30 CPU rack is currently displayed, cursor to the CPU slot and press **Zoom** (**F10**).



```
 1cpu    2iobase 3       4       5       6       7       8       9      10
>
      SERIES 90-20
                  SOFTWARE  CONFIGURATION
      CPU Cat. #: IC692CPU211           SERIES 90-20 CPU 211
      I/O Cat. #:

       IOScan-Stop: NO       Baud Rate  : 19200
       Pwr Up Mode: LAST     Parity     : ODD
       Logic From : RAM      Stop Bits  : 1
       Registers  : RAM      Modem TT   :   0      1/100 Second / Count
       Passwords  : ENABLED  Idle Time  :  10      Seconds

       Chksum Wrds:   4      Sweep Mode : NORMAL
                             Sweep Tmr  : N/A      msec

      << More Config Data Exists: PgDn for Next Page, PgUp for Previous Page >>
                                      OFFLINE
C:\LM90\LESSON                 PRG: LESSON                      CONFIG VALID
REPLACE
```

When the CPU 211 detail screen is first entered, the active field is the *I/O Catalog Number* field. This field is used for the catalog number of the I/O base. (Refer to the information below for configuring the I/O base.)

A. If a different CPU is selected, the screen will display the detail for that CPU. If the new CPU is a CPU 211, the detail screen will be displayed and the active field will be the *I/O Catalog Number* field.

B. If the active field is the *CPU Catalog Number* field, you may enter a new catalog number or edit the current catalog number. You cannot, however, leave this field unless it has a valid entry.

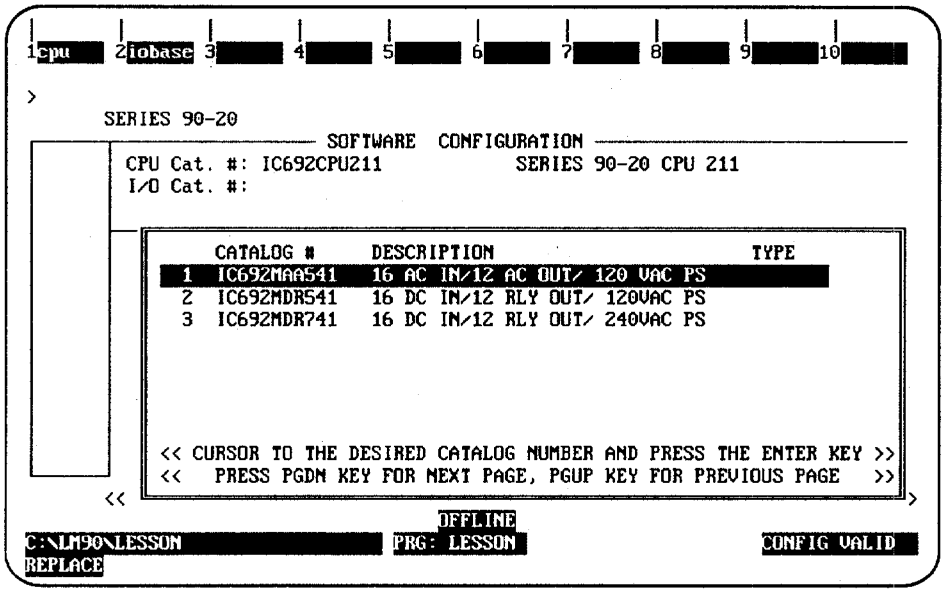
2. The *CPU Parameter* fields may be edited, as described in the following tables.

Parameter	Description
I/O Scan-Stop	Indicates whether the I/O is to be scanned while the PLC is in **STOP** mode. Choices are **YES** or **NO***.
Power-Up Mode	Indicates which state the PLC is to be powered up in. Choices are **RUN**, **STOP**, or **LAST***. **LAST** indicates that the PLC will power up in the same mode it powered down in.
Logic From	Indicates whether the program logic is stored in **RAM***, or in an additional, optional **EEPROM**.
Registers	Indicates whether the source of registers is **RAM*** or **EEPROM**.
Passwords	Indicates whether passwords are **ENABLED*** or **DISABLED**.
Baud Rate	Transmission rate in bits per second. Choices are **300**, **600**, **1200**, **2400**, **4800**, **9600**, or **19200***.
Parity	Type of parity bits added to each word. Choices are **ODD***, **EVEN**, or **NONE**.
Stop Bits	Communication uses at least one stop bit. Slower communication uses two stop bits. Choices are **1*** or **2**.
Modem Turnaround Time	Modem turnaround delay time counts. 1 count = 1/100 second. Choices are **0** to **255** (default = **0**).
Idle Time	Maximum communication idle time from **1** to **60** seconds. (Default = **10** seconds)
Checksum Words	The number of words of user program to be checksummed per sweep. Choices are **0** through **32**, inclusive.
Sweep Mode	Sweep mode can be either **NORMAL SWEEP** mode (**NORMAL***) or **CONSTANT SWEEP** mode (**CNST SWP**). To change the settings, tap the **Down Cursor Movement** key (or **Down Arrow** key) till the cursor rests in the "Sweep Mode:" parameter. Then press the **Tab** key to change the selection from "NORMAL" to "CNST SWP" (if necessary—if you are just changing the time of the Constant Sweep, it may already say "CNST SWP"). Then tap the **Down Arrow** key once to move the cursor to the "Sweep Tmr" parameter and enter the number of milliseconds (from 5 to 200) that you want. To store the changes to the folder, press the **Escape** key twice. Then store the configuration to the PLC and switch the PLC into RUN mode for this change to take effect. In **NORMAL SWEEP** mode, the PLC sweep executes as fast as possible. The overall PLC sweep time depends on the logic program plus the time required to compute the checksum. In **CONSTANT SWEEP** mode, the overall PLC sweep time is fixed.
Sweep Timer	A byte value which can be configured (in 1 millisecond increments) to be any value from **5** to **200** ms. (Default = **100**) Note: The sweep timer is not configurable when the sweep mode is **NORMAL**. In this mode, the timer is set to N/A (Not Applicable), and you cannot cursor to the *Sweep Timer* field.

* Default selection.

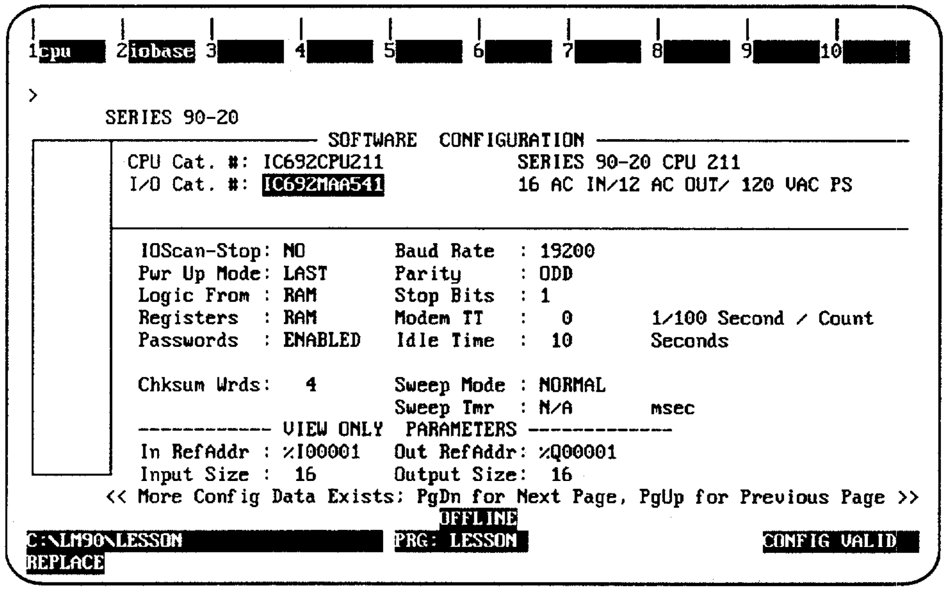
I/O Base Selection for the CPU 211

1. The I/O Base (F2) softkey is used to select a new I/O base for the CPU 211. This key is only displayed after the CPU 211 has been selected.



```
1cpu  2iobase 3      4      5      6      7      8      9      10
>
     SERIES 90-20
                     SOFTWARE  CONFIGURATION
        CPU Cat. #: IC692CPU211            SERIES 90-20 CPU 211
        I/O Cat. #:

              CATALOG #      DESCRIPTION                        TYPE
          1   IC692MAA541    16 AC IN/12 AC OUT/ 120 VAC PS
          2   IC692MDR541    16 DC IN/12 RLY OUT/ 120VAC PS
          3   IC692MDR741    16 DC IN/12 RLY OUT/ 240VAC PS


          << CURSOR TO THE DESIRED CATALOG NUMBER AND PRESS THE ENTER KEY >>
          <<    PRESS PGDN KEY FOR NEXT PAGE, PGUP KEY FOR PREVIOUS PAGE   >>
     <<                                                                      >
                                 OFFLINE
C:\LM90\LESSON                   PRG: LESSON                  CONFIG VALID
REPLACE
```

2. Currently, there are three base modules available (IC692MAA541, IC692MDR541, and IC692MDR741). With the list of catalog numbers and modules displayed, cursor to the desired base and press the **Enter** key to select a new I/O base. The I/O base must be selected before a configuration can be saved or the *I/O Catalog #* field can be left.

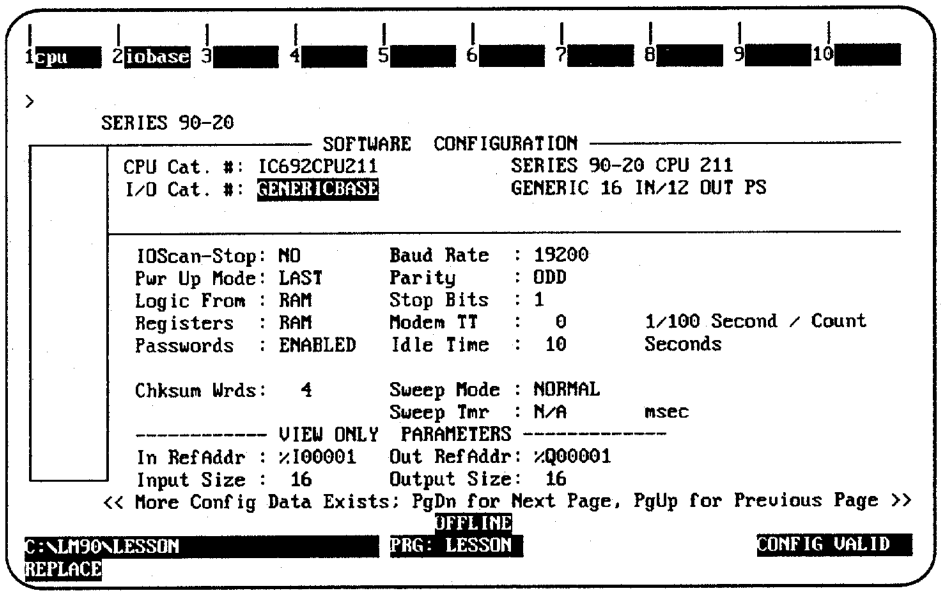


```
1cpu  2iobase 3      4      5      6      7      8      9      10
>
     SERIES 90-20
                     SOFTWARE  CONFIGURATION
        CPU Cat. #: IC692CPU211            SERIES 90-20 CPU 211
        I/O Cat. #: IC692MAA541            16 AC IN/12 AC OUT/ 120 VAC PS

         IOScan-Stop: NO        Baud Rate  : 19200
         Pwr Up Mode: LAST      Parity     : ODD
         Logic From : RAM       Stop Bits  : 1
         Registers  : RAM       Modem TT   :   0      1/100 Second / Count
         Passwords  : ENABLED   Idle Time  :  10      Seconds

         Chksum Wrds:   4       Sweep Mode : NORMAL
                                Sweep Tmr  : N/A      msec
         ------------ VIEW ONLY  PARAMETERS ------------
         In RefAddr : %I00001   Out RefAddr: %Q00001
         Input Size :  16       Output Size:  16
     << More Config Data Exists: PgDn for Next Page, PgUp for Previous Page >>
                                 OFFLINE
C:\LM90\LESSON                   PRG: LESSON                  CONFIG VALID
REPLACE
```

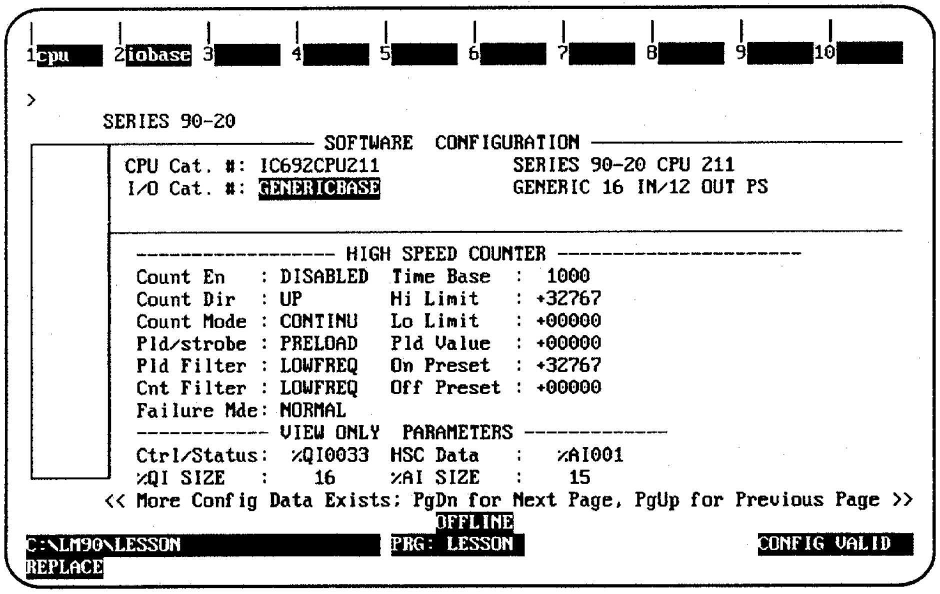
Note

"View Only Parameters" are not editable; therefore, you cannot cursor to them.

3. When a program is loaded from the PLC, if the module ID for each base configured in the system is within the range of 0 to 31 but does not match one of the three I/O bases displayed in the previous screen, the GENERICBASE shown below is displayed. The module ID for GENERICBASE (GENERIC 16 IN/12 OUT POWER SUPPLY) is 31.



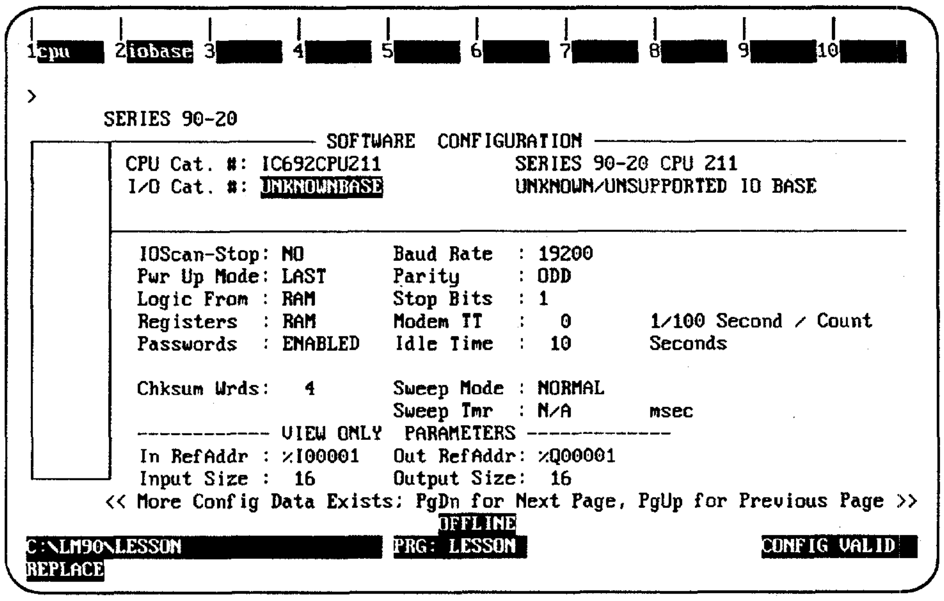
```
1cpu  2iobase 3   4   5   6   7   8   9   10

>
      SERIES 90-20
                   SOFTWARE  CONFIGURATION
       CPU Cat. #: IC692CPU211            SERIES 90-20 CPU 211
       I/O Cat. #: GENERICBASE            GENERIC 16 IN/12 OUT PS

        IOScan-Stop: NO        Baud Rate  : 19200
        Pwr Up Mode: LAST      Parity     : ODD
        Logic From : RAM       Stop Bits  : 1
        Registers  : RAM       Modem TT   :  0      1/100 Second / Count
        Passwords  : ENABLED   Idle Time  :  10     Seconds

        Chksum Wrds:   4       Sweep Mode : NORMAL
                               Sweep Tmr  : N/A     msec
        ------------ VIEW ONLY  PARAMETERS ------------
        In RefAddr : %I00001   Out RefAddr: %Q00001
        Input Size :  16       Output Size:  16
      << More Config Data Exists: PgDn for Next Page, PgUp for Previous Page >>
                                         OFFLINE
C:\LM90\LESSON                       PRG: LESSON              CONFIG VALID
REPLACE
```



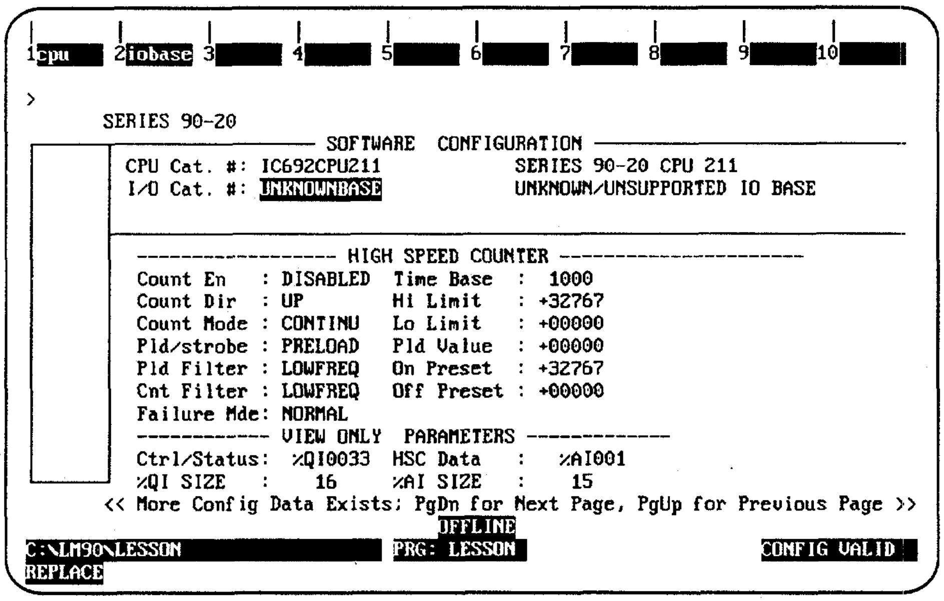
```
1cpu  2iobase 3   4   5   6   7   8   9   10

>
      SERIES 90-20
                   SOFTWARE  CONFIGURATION
       CPU Cat. #: IC692CPU211            SERIES 90-20 CPU 211
       I/O Cat. #: GENERICBASE            GENERIC 16 IN/12 OUT PS

        ----------------- HIGH SPEED COUNTER ---------------------
        Count En    : DISABLED  Time Base  :  1000
        Count Dir   : UP        Hi Limit   : +32767
        Count Mode  : CONTINU   Lo Limit   : +00000
        Pld/strobe  : PRELOAD   Pld Value  : +00000
        Pld Filter  : LOWFREQ   On Preset  : +32767
        Cnt Filter  : LOWFREQ   Off Preset : +00000
        Failure Mde: NORMAL
        ------------ VIEW ONLY  PARAMETERS ------------
        Ctrl/Status: %QI0033    HSC Data   :   %AI001
        %QI SIZE   :    16      %AI SIZE   :    15
      << More Config Data Exists: PgDn for Next Page, PgUp for Previous Page >>
                                         OFFLINE
C:\LM90\LESSON                       PRG: LESSON              CONFIG VALID
REPLACE
```

Refer to section 8, "Configuring an HSC or Embedded HSC," on page 10-47 for an explanation of the parameters and values displayed on this screen.

4. The module ID for UNKNOWNBASE (UNKNOWN/UNSUPPORTED I/O BASE) is 32. If the module ID received from the PLC for the I/O base module is greater than 31, the UNKNOWNBASE base is displayed.



```
 |     |      |     |     |     |     |     |     |     |
1cpu  2iobase 3     4     5     6     7     8     9     10

>
       SERIES 90-20
                     SOFTWARE  CONFIGURATION
         CPU Cat. #: IC692CPU211           SERIES 90-20 CPU 211
         I/O Cat. #: UNKNOWNBASE           UNKNOWN/UNSUPPORTED IO BASE

          IOScan-Stop: NO       Baud Rate  : 19200
          Pwr Up Mode: LAST     Parity     : ODD
          Logic From : RAM      Stop Bits  : 1
          Registers  : RAM      Modem TT   :   0      1/100 Second / Count
          Passwords  : ENABLED  Idle Time  :  10      Seconds

          Chksum Wrds:   4      Sweep Mode : NORMAL
                                Sweep Tmr  : N/A      msec
          ------------ VIEW ONLY  PARAMETERS -------------
          In RefAddr : %I00001  Out RefAddr: %Q00001
          Input Size :  16      Output Size:  16
       << More Config Data Exists; PgDn for Next Page, PgUp for Previous Page >>
                                      OFFLINE
C:\LM90\LESSON                  PRG: LESSON                      CONFIG VALID
REPLACE
```



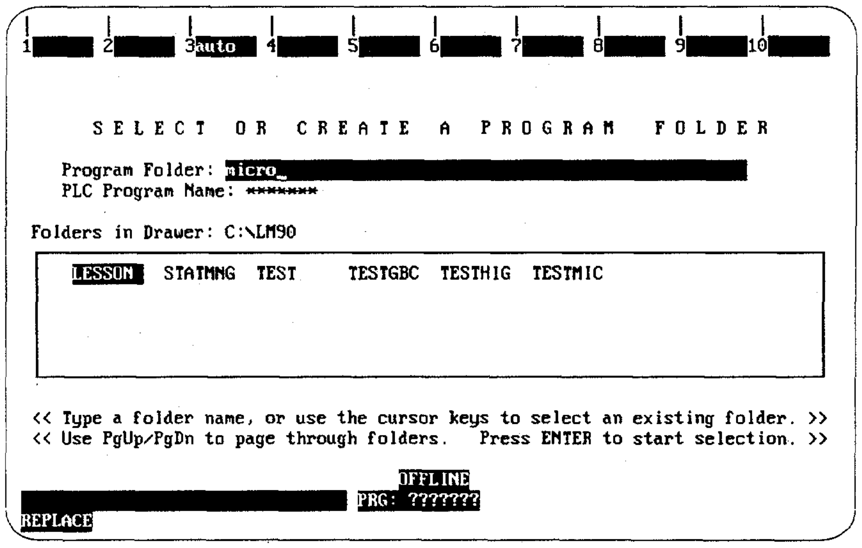
```
 |     |      |     |     |     |     |     |     |     |
1cpu  2iobase 3     4     5     6     7     8     9     10

>
       SERIES 90-20
                     SOFTWARE  CONFIGURATION
         CPU Cat. #: IC692CPU211           SERIES 90-20 CPU 211
         I/O Cat. #: UNKNOWNBASE           UNKNOWN/UNSUPPORTED IO BASE

          ------------------ HIGH SPEED COUNTER ---------------------
          Count En    : DISABLED  Time Base  :  1000
          Count Dir   : UP        Hi Limit   : +32767
          Count Mode  : CONTINU   Lo Limit   : +00000
          Pld/strobe  : PRELOAD   Pld Value  : +00000
          Pld Filter  : LOWFREQ   On Preset  : +32767
          Cnt Filter  : LOWFREQ   Off Preset : +00000
          Failure Mde: NORMAL
          ------------ VIEW ONLY  PARAMETERS -------------
          Ctrl/Status: %QI0033  HSC Data  :   %AI001
          %QI SIZE   :   16     %AI SIZE  :    15
       << More Config Data Exists; PgDn for Next Page, PgUp for Previous Page >>
                                      OFFLINE
C:\LM90\LESSON                  PRG: LESSON                      CONFIG VALID
REPLACE
```

Section 6: Configuring a Micro PLC

A Micro PLC has the CPU, power supply, a high speed counter (except for the AC IN / AC OUT models), inputs and outputs all built into one small device. You can configure the Micro with Logicmaster by following the steps shown below, but also refer to the *Series 90 Micro Programmable Controller User's Manual* (GFK-1065) for additional information.

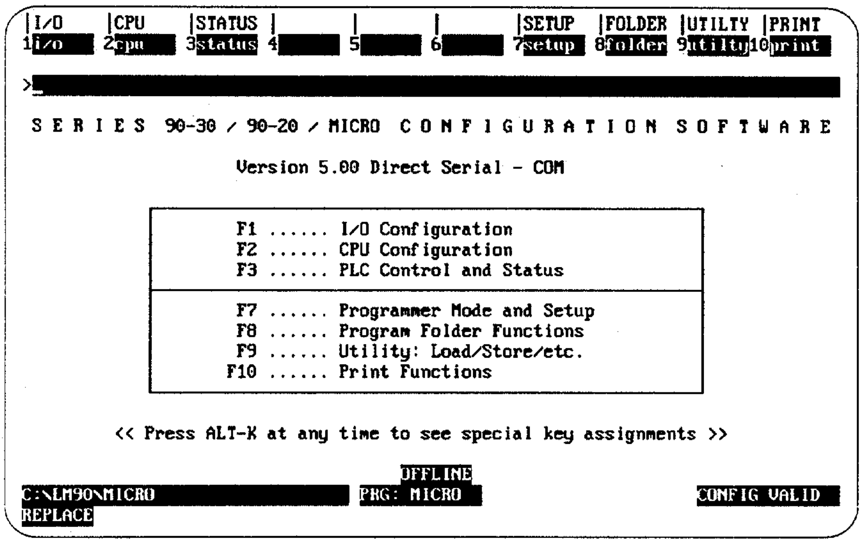
To configure a Micro PLC, follow these steps:

1. Press the **`Micro`** softkey (**`Shift-F1`**) from the Logicmaster 90-30 Main Menu (i.e., the first screen you see after entering Logicmaster). This highlights the word *Micro* at the top of the Logicmaster display showing you that it is selected.
2. Press the **`Config`** softkey (**`F2`**). After a short pause (during which the Logicmaster copyright screen appears), your screen display will change to one similar to the one shown below.



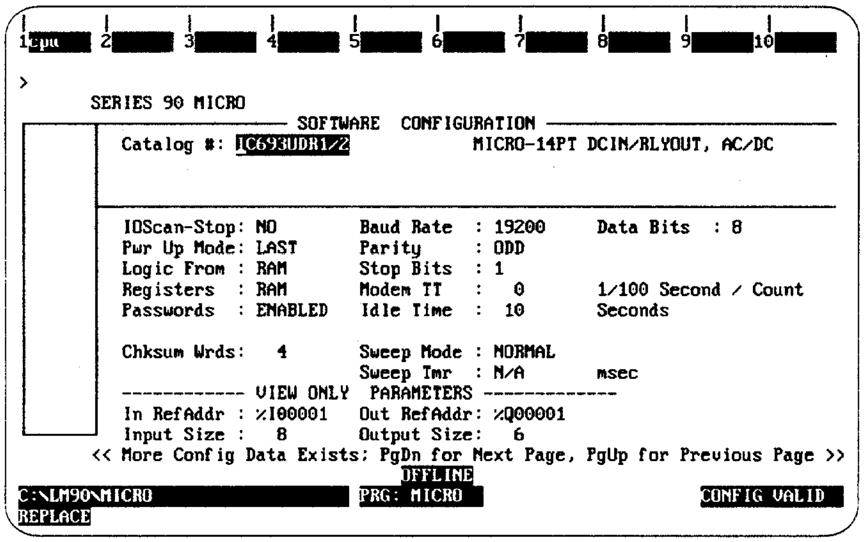
```
1      2      3auto  4      5      6      7      8      9      10

        S E L E C T   O R   C R E A T E   A   P R O G R A M   F O L D E R

     Program Folder: micro_
     PLC Program Name: ********

  Folders in Drawer: C:\LM90

     LESSON   STATMNG   TEST     TESTGBC   TESTHIG   TESTMIC

  << Type a folder name, or use the cursor keys to select an existing folder. >>
  << Use PgUp/PgDn to page through folders.   Press ENTER to start selection. >>

                                              OFFLINE
                                              PRG: ???????
REPLACE
```

3. Type the name you want to call your new folder. (In the example shown above, "micro" was used.) Then press **`Enter`**. Logicmaster will then ask you if you want to "create new folder?" Type **`Y`** for **`Yes`**. Your screen will look like the one shown below.



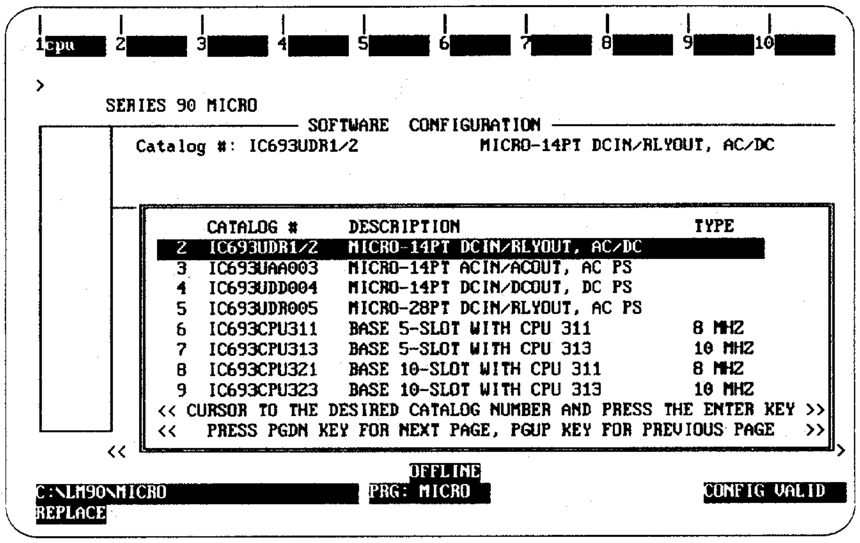
```
|I/O    |CPU    |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1i/o    2cpu    3status 4       5       6       7setup  8folder 9utilty10print
>_

S E R I E S  90-30 / 90-20 / MICRO  C O N F I G U R A T I O N  S O F T W A R E

                  Version 5.00 Direct Serial - COM

                  F1 ...... I/O Configuration
                  F2 ...... CPU Configuration
                  F3 ...... PLC Control and Status

                  F7 ...... Programmer Mode and Setup
                  F8 ...... Program Folder Functions
                  F9 ...... Utility: Load/Store/etc.
                 F10 ...... Print Functions

        << Press ALT-K at any time to see special key assignments >>
                                     OFFLINE
C:\LM90\MICRO                        PRG: MICRO               CONFIG VALID
REPLACE
```

4. Press the **I/O** softkey (**F1**). The following screen will appear.



```
|       |       |       |       |       |       |       |       |       |
1cpu    2       3       4       5       6       7       8       9      10
>
      SERIES 90 MICRO
                       SOFTWARE  CONFIGURATION
         Catalog #: IC693UDR1/2            MICRO-14PT DCIN/RLYOUT, AC/DC

         IOScan-Stop: NO        Baud Rate  : 19200     Data Bits  : 8
         Pwr Up Mode: LAST      Parity     : ODD
         Logic From : RAM       Stop Bits  : 1
         Registers  : RAM       Modem TT   :   0       1/100 Second / Count
         Passwords  : ENABLED   Idle Time  :  10       Seconds

         Chksum Wrds:   4       Sweep Mode : NORMAL
                                Sweep Tmr  : N/A       msec
         ------------ VIEW ONLY  PARAMETERS -------------
         In RefAddr : %I00001   Out RefAddr: %Q00001
         Input Size :   8       Output Size:   6
      << More Config Data Exists: PgDn for Next Page, PgUp for Previous Page >>
                                     OFFLINE
C:\LM90\MICRO                        PRG: MICRO               CONFIG VALID
REPLACE
```

5. If the Micro you are installing ***has*** the same Catalog # as the default selection, *then skip to step 7* discussed on the next page. If the Micro you are installing ***does not*** have the same Catalog # as the default selection, then press the **CPU** softkey (**F1**). The following screen will appear.

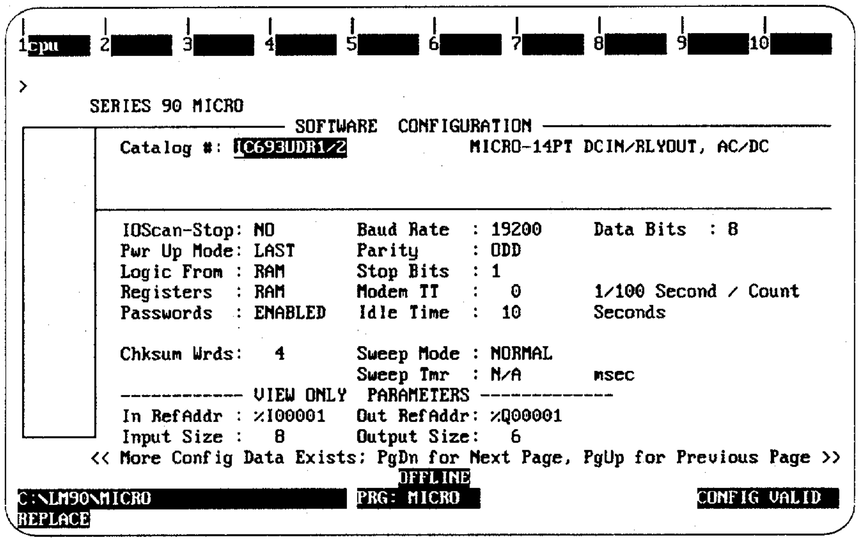


```
 1cpu   2     3     4     5     6     7     8     9     10
>
     SERIES 90 MICRO
                         SOFTWARE  CONFIGURATION
        Catalog #: IC693UDR1/2           MICRO-14PT DCIN/RLYOUT, AC/DC

           CATALOG #     DESCRIPTION                       TYPE
         2  IC693UDR1/2   MICRO-14PT DCIN/RLYOUT, AC/DC
         3  IC693UAA003   MICRO-14PT ACIN/ACOUT, AC PS
         4  IC693UDD004   MICRO-14PT DCIN/DCOUT, DC PS
         5  IC693UDR005   MICRO-28PT DCIN/RLYOUT, AC PS
         6  IC693CPU311   BASE 5-SLOT WITH CPU 311          8 MHZ
         7  IC693CPU313   BASE 5-SLOT WITH CPU 313          10 MHZ
         8  IC693CPU321   BASE 10-SLOT WITH CPU 311         8 MHZ
         9  IC693CPU323   BASE 10-SLOT WITH CPU 313         10 MHZ
        << CURSOR TO THE DESIRED CATALOG NUMBER AND PRESS THE ENTER KEY >>
        <<   PRESS PGDN KEY FOR NEXT PAGE, PGUP KEY FOR PREVIOUS PAGE   >>
      <<                                                                >
                                  OFFLINE
C:\LM90\MICRO                   PRG: MICRO                   CONFIG VALID
REPLACE
```

Note

Step 6 is **only necessary** *if* the Micro you are installing *does not* have the same Catalog # as the default selection.

6. Move the cursor up or down using the **Cursor Movement** (or **Arrow**) keys to the correct selection. (When scrolling through the list, you may notice that other options include switching to a 90-20 or a 90-30.) When you have your cursor on the correct Catalog #, press **Enter** which will take you back to the screen shown on the top of this page.



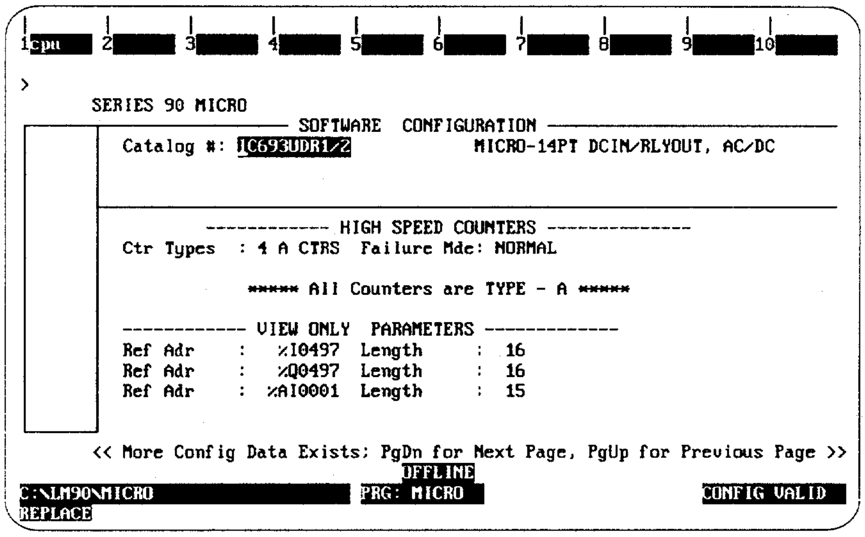
```
 1cpu   2     3     4     5     6     7     8     9     10
>
     SERIES 90 MICRO
                         SOFTWARE  CONFIGURATION
        Catalog #: IC693UDR1/2           MICRO-14PT DCIN/RLYOUT, AC/DC

        IOScan-Stop: NO        Baud Rate  : 19200     Data Bits  : 8
        Pwr Up Mode: LAST      Parity     : ODD
        Logic From : RAM       Stop Bits  : 1
        Registers  : RAM       Modem TT   :   0       1/100 Second / Count
        Passwords  : ENABLED   Idle Time  :  10       Seconds

        Chksum Wrds:    4      Sweep Mode : NORMAL
                               Sweep Tmr  : N/A       msec
        ------------ VIEW ONLY  PARAMETERS -------------
        In RefAddr : %I00001   Out RefAddr: %Q00001
        Input Size :   8       Output Size:   6
     << More Config Data Exists: PgDn for Next Page, PgUp for Previous Page >>
                                  OFFLINE
C:\LM90\MICRO                   PRG: MICRO                   CONFIG VALID
REPLACE
```

Note

Parameter descriptions for each of the following screens is presented in the Parameter Description tables beginning on page 10-34.

7. On the screen shown above, you can move your cursor from field to field by pressing the **Cursor Movement** (or **Arrow**) keys. When you are in the field you want to modify, you can either type in your choice or press the **Tab** key to scroll through the

available selections (or **Shift-Tab** to reverse the direction of the scrolling). From this screen, press the **Page Down** key to take you to the screen shown below.


```
 1 cpu  2  3  4  5  6  7  8  9  10
>
      SERIES 90 MICRO
                          SOFTWARE  CONFIGURATION
          Catalog #: IC693UDR1/2          MICRO-14PT DCIN/RLYOUT, AC/DC

                    ------------ HIGH SPEED COUNTERS --------------
          Ctr Types  : 4 A CTRS  Failure Mde: NORMAL

                        ***** All Counters are TYPE - A *****

          ------------ VIEW ONLY  PARAMETERS -------------
          Ref Adr    :   %I0497  Length     :  16
          Ref Adr    :   %Q0497  Length     :  16
          Ref Adr    :  %AI0001  Length     :  15

      << More Config Data Exists: PgDn for Next Page, PgUp for Previous Page >>
                                            OFFLINE
C:\LM90\MICRO                          PRG: MICRO                 CONFIG VALID
REPLACE
```

Notes:

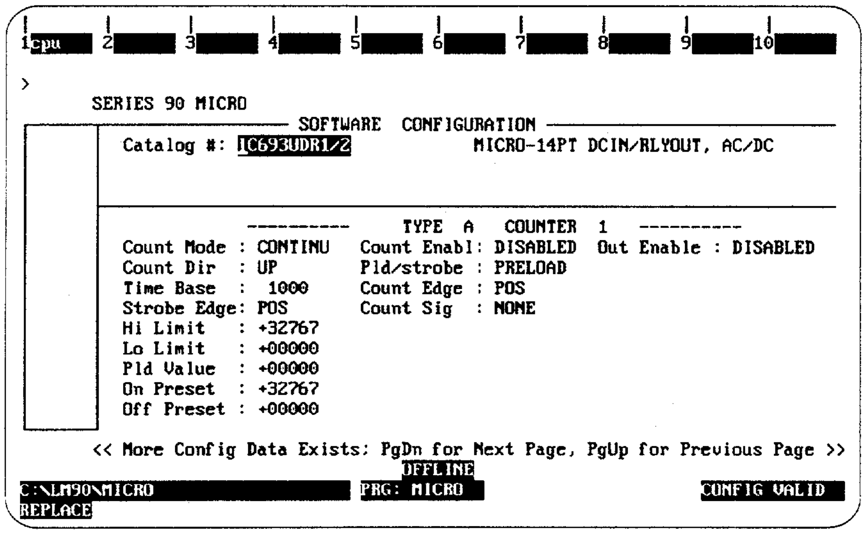
A. %AQ reference addresses will display for DC OUT models only.

B. The High Speed Counter screens will not appear for AC IN / AC OUT models (since they do not have High Speed Counters).

Note

If you are configuring the HSC (high speed counter) mode B1–3, A4, move the cursor to the **Ctr Types** field, then press the **Tab** key once to toggle the default selection to B1–3, A4. Counters B1–3 are then configured on one screen, A4 on another, pressing the **Page Down** key to advance screens.

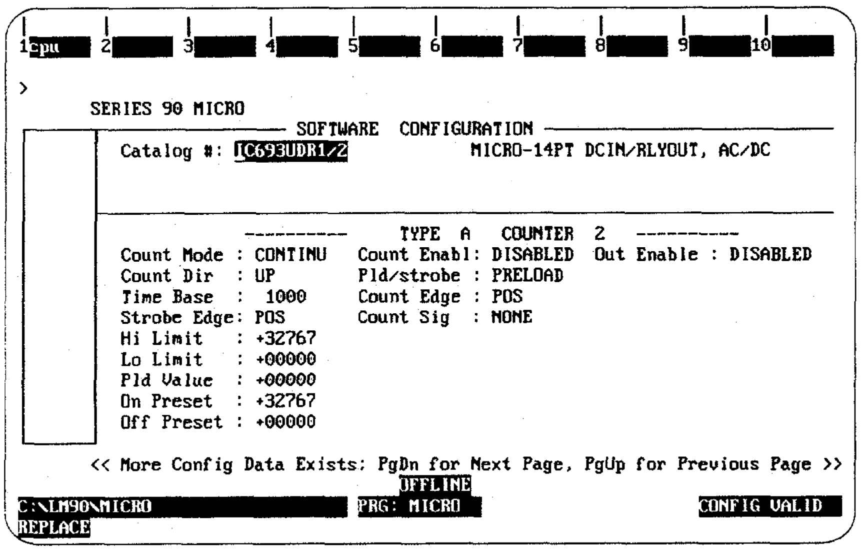
8. You can move your cursor from field to field by pressing the **Cursor Movement** (or **Arrow**) keys. When you are in the field you want to modify, you can either type in your choice or press the **Tab** key to scroll through the available selections (or **Shift-Tab** to reverse the direction of the scrolling). From this screen, press the **Page Down** key to take you to the screen shown below.


```
 1 cpu  2  3  4  5  6  7  8  9  10
>
      SERIES 90 MICRO
                          SOFTWARE  CONFIGURATION
          Catalog #: IC693UDR1/2          MICRO-14PT DCIN/RLYOUT, AC/DC

                      ----------      TYPE  A   COUNTER  1  ----------
          Count Mode  : CONTINU   Count Enabl: DISABLED  Out Enable : DISABLED
          Count Dir   : UP        Pld/strobe : PRELOAD
          Time Base   :  1000     Count Edge : POS
          Strobe Edge: POS        Count Sig  : NONE
          Hi Limit    : +32767
          Lo Limit    : +00000
          Pld Value   : +00000
          On Preset   : +32767
          Off Preset  : +00000

      << More Config Data Exists: PgDn for Next Page, PgUp for Previous Page >>
                                            OFFLINE
C:\LM90\MICRO                          PRG: MICRO                 CONFIG VALID
REPLACE
```

Note

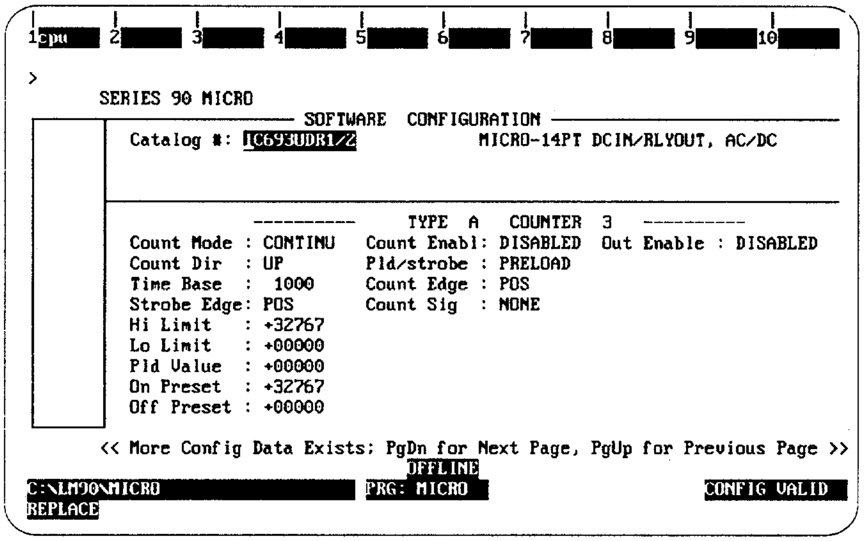
Additional fields, **PMW Out %Q1** and **Pul Out %Q2**, will display on DC OUT models. *This applies to all of the following screens.*

9. You can move your cursor from field to field by pressing the **Cursor Movement** (or **Arrow**) keys. When you are in the field you want to modify, you can either type in your choice or press the **Tab** key to scroll through the available selections (or **Shift-Tab** to reverse the direction of the scrolling). From this screen, press the **Page Down** key to take you to the screen shown below.



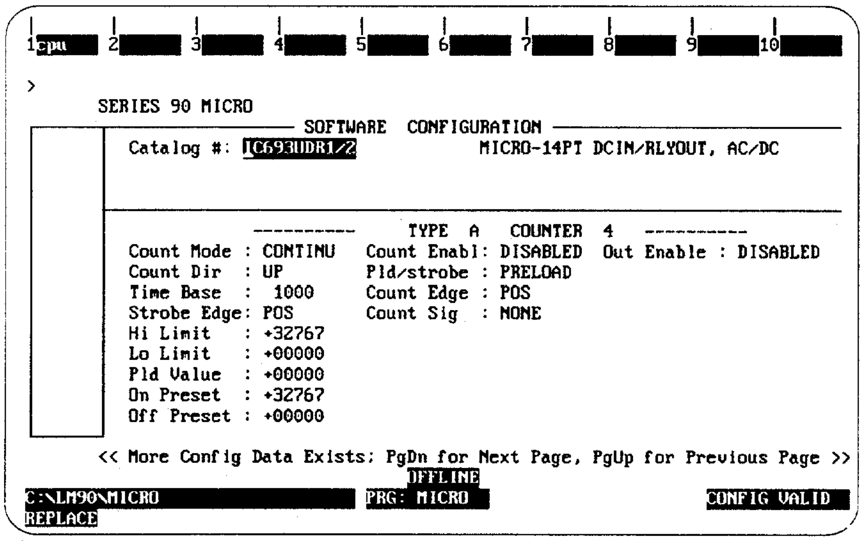
```
1cpu  2  3  4  5  6  7  8  9  10
>
      SERIES 90 MICRO
                    ---- SOFTWARE  CONFIGURATION ----
         Catalog #: IC693UDR1/2            MICRO-14PT DCIN/RLYOUT, AC/DC

                  ----------   TYPE  A   COUNTER  2  ----------
         Count Mode  : CONTINU   Count Enabl: DISABLED  Out Enable : DISABLED
         Count Dir   : UP        Pld/strobe : PRELOAD
         Time Base   :  1000     Count Edge : POS
         Strobe Edge: POS        Count Sig  : NONE
         Hi Limit    : +32767
         Lo Limit    : +00000
         Pld Value   : +00000
         On Preset   : +32767
         Off Preset  : +00000
      << More Config Data Exists: PgDn for Next Page, PgUp for Previous Page >>
                                   OFFLINE
C:\LM90\MICRO                     PRG: MICRO                      CONFIG VALID
REPLACE
```

10. You can move your cursor from field to field by pressing the **Cursor Movement** (or **Arrow**) keys. When you are in the field you want to modify, you can either type in your choice or press the **Tab** key to scroll through the available selections (or **Shift-Tab** to reverse the direction of the scrolling). From this screen, press the **Page Down** key to take you to the screen shown below.



```
1cpu  2     3     4     5     6     7     8     9     10
>
     SERIES 90 MICRO
               --------- SOFTWARE  CONFIGURATION ---------
       Catalog #: IC693UDR1/2          MICRO-14PT DCIN/RLYOUT, AC/DC

                ----------  TYPE  A   COUNTER  3  ----------
       Count Mode  : CONTINU  Count Enabl: DISABLED  Out Enable : DISABLED
       Count Dir   : UP       Pld/strobe : PRELOAD
       Time Base   :  1000    Count Edge : POS
       Strobe Edge: POS       Count Sig  : NONE
       Hi Limit    : +32767
       Lo Limit    : +00000
       Pld Value   : +00000
       On Preset   : +32767
       Off Preset  : +00000

     << More Config Data Exists; PgDn for Next Page, PgUp for Previous Page >>
                                  OFFLINE
C:\LM90\MICRO                   PRG: MICRO                  CONFIG VALID
REPLACE
```

11. Use the same techniques to set up this screen as the previous ones; i.e., you can move your cursor from field to field by pressing the **Cursor Movement** (or **Arrow**) keys. When you are in the field you want to modify, you can either type in your choice or press the **Tab** key to scroll through the available selections (or **Shift-Tab** to reverse the direction of the scrolling). From this screen, press the **Page Down** key to take you to the screen shown below.



```
1cpu  2     3     4     5     6     7     8     9     10
>
     SERIES 90 MICRO
               --------- SOFTWARE  CONFIGURATION ---------
       Catalog #: IC693UDR1/2          MICRO-14PT DCIN/RLYOUT, AC/DC

                ----------  TYPE  A   COUNTER  4  ----------
       Count Mode  : CONTINU  Count Enabl: DISABLED  Out Enable : DISABLED
       Count Dir   : UP       Pld/strobe : PRELOAD
       Time Base   :  1000    Count Edge : POS
       Strobe Edge: POS       Count Sig  : NONE
       Hi Limit    : +32767
       Lo Limit    : +00000
       Pld Value   : +00000
       On Preset   : +32767
       Off Preset  : +00000

     << More Config Data Exists; PgDn for Next Page, PgUp for Previous Page >>
                                  OFFLINE
C:\LM90\MICRO                   PRG: MICRO                  CONFIG VALID
REPLACE
```

12. After you have set up the final counter, press the **Escape** key to save your configuration. This will also return you to the main menu of the Configurator package.

CPU PARAMETERS

Parameter	Type	Description	Possible Values	Default
IOScan−Stop	Editable	Indicates whether I/O is to be scanned while the PLC is in STOP mode.	YES, NO	NO
PWR Up Mode	Editable	Indicates what the power up mode will be.	LAST, STOP, RUN	LAST
Data Bits	Editable	Specifies whether the CPU recognizes 7-bit or 8-bit characters. (SNP ***requires*** 8-bit.)	7, 8	8
Registers	Editable	Indicates whether the source of register data is RAM or PROM when the PLC is powered up.	RAM, PROM	RAM
Passwords	Editable	Indicates whether the password feature is ENABLED or DISABLED.	ENABLED, DISABLED	ENABLED
Baud Rate	Editable	Indicates the transmission rate of data through the port (in bits per second).	300, 600, 1200, 2400, 4800, 9600, 19200	19200
Parity	Editable	Indicates whether parity is added to words.	ODD, EVEN, NONE	ODD
Stop Bits	Editable	Indicates the number of stop bits used in transmission.	1, 2	1
Modem TT	Editable	Indicates the time required for the modem to start data transmission after receiving the transmit request.	0−255	0
Idle Time	Editable	Indicates the time the CPU should wait for the next message to be received from the programming device before it assumes that the programming device has failed and proceeds to its base state.	1−60	10
In Ref Adr	Non-Editable	Discrete input reference.	%I00001	%I00001
Input Size	Non-Editable	Discrete input size.	8	8
Out Ref Adr	Non-Editable	Discrete output reference.	%Q00001	%Q00001
Output Size	Non-Editable	Discrete output size.	6	6
Logic From	Editable	Indicates whether the source of logic is RAM or PROM when the PLC is powered up.	RAM, PROM	RAM
Chksum Wrds	Editable	Indicates the number of words of the user program to be used as input to the check-sum algorithm each sweep.	4−32	4

CPU PARAMETERS (continued)

Parameter	Type	Description	Possible Values	Default
Sweep Mode	Editable	Indicates the sweep mode. Normal sweep mode means that the sweep will run until it is complete. Constant sweep mode means that the sweep will always run for the time specified by the user.	NORMAL, CNST SWP	NORMAL
Sweep Tmr	Editable when Sweep Mode is CNST SWP; otherwise, Non-Editable.	Indicates the constant sweep time	5–200 when Sweep Mode is CNST SWP; N/A when Sweep Mode is NORMAL.	N/A

HSC PARAMETERS

Parameter	Type	Description	Possible Values	Default
Ctr Types	Editable	The counter types of the counters of the HSC.	4 A CTRS, B1/2A3/4	4 A CTRS
Ref Adr	Non-Editable	Discrete input reference.	%I00497	%I00497
Length	Non-Editable	Discrete input size.	16	16
Ref Adr	Non-Editable	Discrete output reference.	%Q00497	%Q00497
Length	Non-Editable	Discrete output size.	16	16
Ref Adr	Non-Editable	Analog input reference.	%AI0001	%AI0001
Length	Non-Editable	Analog input size.	15	15
Ref Adr (Displayed only for DC OUTPUT models)	Non-Editable	Analog output reference.	%AQ0002	%AQ0002
Length (Displayed only for DC OUTPUT models)	Non-Editable	Analog output size.	8	8
Ref Adr (Displayed only for DC OUTPUT models)	Non-Editable	Analog output reference.	%IQ0497 %Q0497 %AI0001	%IQ0497 %Q0497 %AI0001
Length (Displayed only for DC OUTPUT models)	Non-Editable	Analog output size.	6	6
Count Mode	Editable	Count mode determines the accumulator count method.	CONTINUOUS, SINGLE–SHOT	CONTINUOUS
Count Dir	Editable	The count direction.	UP, DOWN	UP
Time Base	Editable	The span of time which can be used to measure the rate of counting.	10–65535	1000
Hi Limit	Editable	The high limit of the counter. (1) The Hi Limit must not be lower than the Lo Limit; (2) in addition, the Pld Value, ON Preset, and OFF Preset values cannot be higher than the Hi Limit.	–32768 to +32767	+32767

Lo Limit	Editable	The low limit of the counter. (1) The Lo Limit must not be higher than the Hi Limit; (2) in addition, the Pld Value, ON Preset, and OFF Preset values cannot be lower than the Lo Limit.	−32768 to +32767	+00000
Strobe Edge	Editable	Strobe inputs can be configured to trigger on either a POSITIVE or NEGATIVE edge.	POS, NEG	POS
Pld Value	Editable	The starting count value of the counter. (1) The Pld Value must be between the configured Hi and Lo Limits.	−32768 to +32767	+00000

HSC PARAMETERS (continued)

Parameter	Type	Description	Possible Values	Default
ON Preset	Editable	The ON preset of the counter. (1) The ON Preset must be between the configured Hi and Lo Limits.	−32768 to +32767	+32767
OFF Preset	Editable	The OFF preset of the counter. (1) The OFF Preset must be between the configured Hi and Lo Limits.	−32768 to +32767	+00000
Count Enabl	Editable	Indicates whether or not the counter is enabled. (1) For counter 1, PWM Out %Q1 and Pul Out %Q1 fields must be DISABLED when its "Count Enabl" is ENABLED; Count Enabl must be DISABLED when either PWM Out %Q1 or Pul Out %Q1 is ENABLED. (2) For counter 2, PWM Out %Q2 and Pul Out %Q3 fields must be DISABLED when its Count Enabl is ENABLED; Count Enabl must be DISABLED when either PWM Out %Q2 or Pul Out %Q3 is ENABLED. (3) For counter 3, PWM Out %Q3 and Pul Out %Q5 fields must be DISABLED when its Count Enabl is ENABLED; Count Enabl must be DISABLED when either PWM Out %Q3 or Pul Out %Q5 is ENABLED. (4) For counter 4, PWM Out %Q4 must be DISABLED when its Count Enabl is ENABLED; Count Enabl must be DISABLED when PWM Out %Q4 is ENABLED. (5) For all counters, Out Enable must be DISABLED when Count Enable is DISABLED.	ENABLED, DISABLED	DISABLED
Pld/Strobe	Editable	Indicates which register should be used when an interrupt is detected on the preload/strobe input.	PRELOAD, STROBE	PRELOAD
Count Edge	Editable	Count inputs can be configured to trigger on either a POSITIVE or NEGATIVE edge.	POS, NEG	POS
Failure Mde	Editable	The state of the outputs when there is a loss of PLC.	NORMAL, FORCEOFF, HOLDLAST	NORMAL
Count Sig	Non-Editable.	The count signal mode. For all counters, Count SIg will be NONE when the counter is an A-TYPE counter; Count Sig will be AQUADB when the counter is a B-TYPE counter.	NONE, AQUADB	Dependent on the counter type.
Out Enable	Non-Editable	Indicates whether or not the HSC controls the counter output. (1) For all counters, Out Enable will be DISABLED when Count Enabl is DISABLED.	ENABLED, DISABLED	DISABLED

HSC PARAMETERS (continued)

Parameter	Type	Description	Possible Values	Default
PWM Out %Q1 (Displayed only for DC OUTPUT models)	Editable	Indicates whether or not counter 1 outputs will be controlled by %Q1, and also whether or not these outputs will be PWM outputs. (1) When PWM Out %Q1 is ENABLED, Count Enabl and Pul Out %Q1 both have to be DISABLED; (2) when either Pul Out %Q1 or Count Enabl is ENABLED, PWM Out %Q1 has to be DISABLED.	ENABLED, DISABLED	DISABLED
Pul Out %Q1 (Displayed only for DC OUTPUT models)	Editable	Indicates whether or not counter 1 outputs will be controlled by %Q1, and also whether or not these outputs will be Pulse Train outputs. (1) When Pul Out %Q1 is ENABLED, Count Enabl and PWM Out %Q1 have to be DISABLED; (2) when either PWM Out %Q1 or Count Enabl is ENABLED, Pul Out %Q1 has to be DISABLED.	ENABLED, DISABLED	DISABLED
PWM Out %Q2 (Displayed only for DC OUTPUT models)	Editable	Indicates whether or not counter 2 outputs will be controlled by %Q2, and also whether or not these outputs will be PWM outputs. (1) When PWM Out %Q2 is ENABLED, Count Enabl and Pul Out %Q3 have to be DISABLED; (2) when either Pul Out %Q3 or Count Enabl is ENABLED, PWM Out %Q2 has to be DISABLED.	ENABLED, DISABLED	DISABLED
Pul Out %Q3 (Displayed only for DC OUTPUT models)	Editable	Indicates whether or not counter 2 outputs will be controlled by %Q3, and also whether or not these outputs will be Pulse Train outputs. (1) When Pul Out %Q3 is ENABLED, Count Enabl and PWM Out %Q2 have to be DISABLED; (2) when either PWM Out %Q2 or Count Enabl is ENABLED, Pul Out %Q3 has to be DISABLED.	ENABLED, DISABLED	DISABLED
PWM Out %Q3 (Displayed only for DC OUTPUT models)	Editable	Indicates whether or not counter 3 outputs will be controlled by %Q3, and also whether or not these outputs will be PWM outputs. (1) When PWM Out %Q3 is ENABLED, Count Enabl and Pul Out %Q5 have to be DISABLED; (2) when either Pul Out %Q5 or Count Enabl is ENABLED, PWM Out %Q3 has to be DISABLED.	ENABLED, DISABLED	DISABLED

HSC PARAMETERS (continued)

Parameter	Type	Description	Possible Values	Default
Pul Out %Q5 (Displayed only for DC OUTPUT models)	Editable	Indicates whether or not counter 3 outputs will be controlled by %Q5, and also whether or not these outputs will be Pulse Train outputs. (1) When Pul Out %Q5 is ENABLED, Count Enabl and PWM Out %Q3 have to be DISABLED; (2) when either PWM Out %Q3 or Count Enabl is ENABLED, Pul Out %Q5 has to be DISABLED.	ENABLED, DISABLED	DISABLED
PWM Out %Q4 (Displayed only for DC OUTPUT models)	Editable	Indicates whether or not counter 4 outputs will be controlled by %Q4, and also whether or not these outputs will be PWM outputs. (1) When PWM Out %Q4 is ENABLED, Count Enabl has to be DISABLED; (2) when Count Enabl is ENABLED, PWM Out %Q4 has to be DISABLED.	ENABLED, DISABLED	DISABLED

PLC Memory Configuration Limits

Memory Type	Memory Limit
%R	256 words
%AI	128 words
%AQ	128 words
%I	512 bits
%Q	512 bits
%G	1280 bits
%M	1024 bits
%T	256 bits
%S	128 bits

Note

These limits are specific to Micro PLCs.

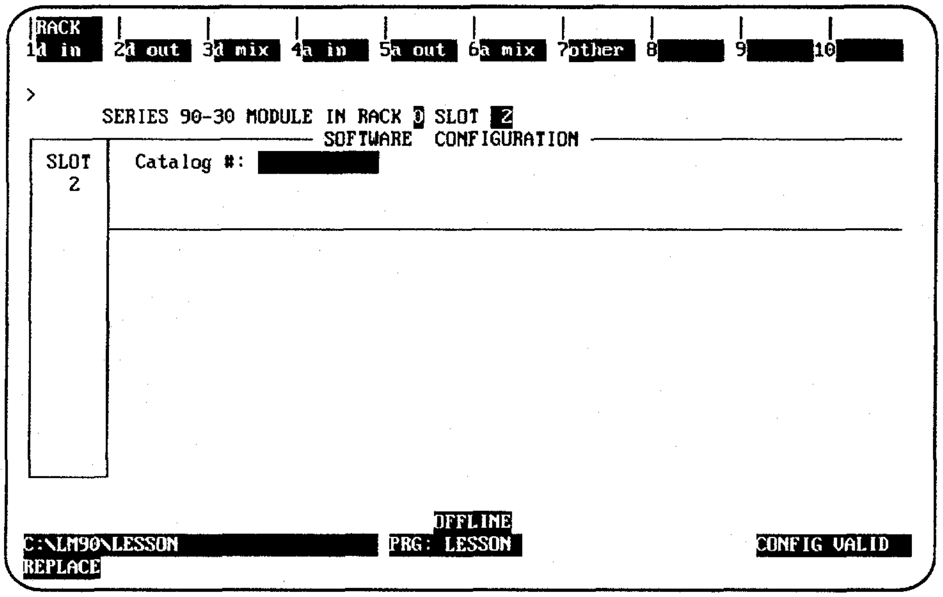
Configuration Rules and Miscellaneous Information

1. When replacing the CPU (not a CPU211) of a non-default configuration with a SERIES 90 Micro, the additional prompt, "Existing I/O Configuration will be lost, Continue REPLACE ? **(Y/N)**" will be displayed. Confirming the prompt will cause the replacement to occur; otherwise, the replacement will not occur.
2. When replacing a CPU211 with a SERIES 90 Micro, no additional prompts will be displayed.
3. When replacing a SERIES 90 Micro with another CPU (not a CPU211), the resulting configuration will consist of the following: (1) a PS in rack 0, slot 0; (2) the new CPU in rack 0, slot 1; (3) an 8-pt. Discrete Input module in rack 0, slot 2; (4) an 8-pt Discrete Output module in rack 0, slot 3, (5) a 9030 HSC in rack 0, slot 4, and (6) all other racks unconfigured.
4. If you select the **MICRO** softkey from the Logicmaster 90 main menu and then select an existing folder that contained either a 9030 or the 9020 configuration, then that existing configuration will be displayed when the **I/O** softkey is pressed from the Configurator main menu.
5. If you select either the **90-20** or **90-30** softkey from the Logicmaster 90 main menu and then select an existing folder that contained a SERIES 90 Micro configuration, then that existing configuration will be displayed when the **I/O** softkey is pressed from the Configurator main menu.

Section 7: Configuring 90-30 I/O Modules

To configure a 90-30 I/O module:

1. Move the cursor to the slot where the module will be located and press Module 30 I/O (**F1**).



```
|RACK    |        |        |        |        |        |        |        |        |
1d in   2d out   3d mix   4a in    5a out   6a mix   7other   8        9        10
>
      SERIES 90-30 MODULE IN RACK 0 SLOT 2
                          SOFTWARE  CONFIGURATION
 SLOT  | Catalog #:
  2    |

                                          OFFLINE
C:\LM90\LESSON                            PRG: LESSON                 CONFIG VALID
REPLACE
```

2. Select the module type by pressing the corresponding function key listed below.

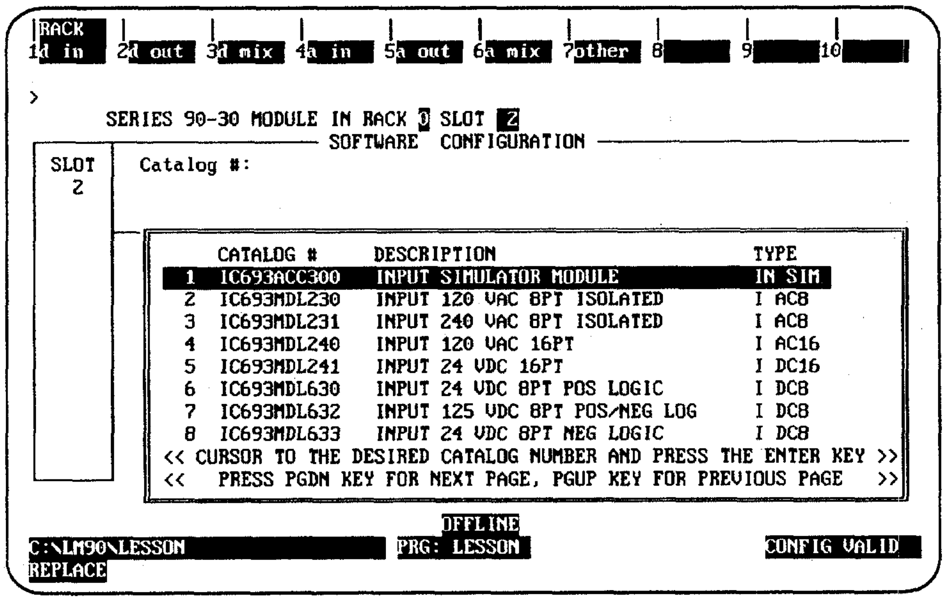
Function Key	Module Type	Description
F1	d in	Discrete input module.
F2	d out	Discrete output module.
F3	d mix	Discrete mixed module.
F4	a in	Analog input module.
F5	a out	Analog output module.
F6	a mix	Analog mixed module
F7	other	Other modules if available (see Note below)

Note

Other module types are currently unavailable. This key is reserved for possible future use.

Press **ALT-H** to display help information for further explanations.

3. The next screen lists the catalog numbers and available modules of the type you selected. For example, when **F1** is pressed, the following screen is displayed.



```
|RACK  |      |      |      |      |      |      |      |      |
1 i in 2 i out 3 i mix 4a in 5a out 6a mix 7other 8      9      10

>
        SERIES 90-30 MODULE IN RACK 0 SLOT 2
                          SOFTWARE CONFIGURATION
 SLOT   Catalog #:
  2

              CATALOG #     DESCRIPTION                       TYPE
           1  IC693ACC300   INPUT SIMULATOR MODULE            IN SIM
           2  IC693MDL230   INPUT 120 VAC 8PT ISOLATED        I AC8
           3  IC693MDL231   INPUT 240 VAC 8PT ISOLATED        I AC8
           4  IC693MDL240   INPUT 120 VAC 16PT                I AC16
           5  IC693MDL241   INPUT 24 VDC 16PT                 I DC16
           6  IC693MDL630   INPUT 24 VDC 8PT POS LOGIC        I DC8
           7  IC693MDL632   INPUT 125 VDC 8PT POS/NEG LOG     I DC8
           8  IC693MDL633   INPUT 24 VDC 8PT NEG LOGIC        I DC8
         << CURSOR TO THE DESIRED CATALOG NUMBER AND PRESS THE ENTER KEY >>
         <<   PRESS PGDN KEY FOR NEXT PAGE, PGUP KEY FOR PREVIOUS PAGE   >>

                                 OFFLINE
C:\LM90\LESSON                   PRG: LESSON                CONFIG VALID
REPLACE
```

4. Select a module by moving the cursor down to any entry. Additional entries may be displayed by pressing the **Page Down** key. Then, press the **Enter** key.

5. Move the cursor to the module's reference address. The next highest available reference address is automatically configured.

   If you want to use a different reference address, enter it here. For a discrete module, the reference address must begin on a byte boundary. (A byte boundary is a reference address which is a multiple of 8 + 1; e.g., 1, 9, 17, or 25.) The software will automatically adjust an incorrect entry to begin on a byte boundary. A message is displayed when the highest available address has been assigned.

6. Press **Rack (Shift-F1)** or the **Escape** key to return to the rack display.

Note

12-point modules do not use consecutive address bits in PLC memory, but they use bits 1 through 6 and 9 through 14. For example, if %Q0001 is assigned to a 12-point output module, %Q0001 through %Q0006 and %Q0009 through %Q0014 will be associated with the output point.

Configuring Generic I/O Modules

If a specific catalog number does not exist for a particular module, the configuration can assign a generic reference address to that module.

The following table lists the catalog numbers to be used to configure modules generically and the function key that would be pressed in order to select a particular generic module.

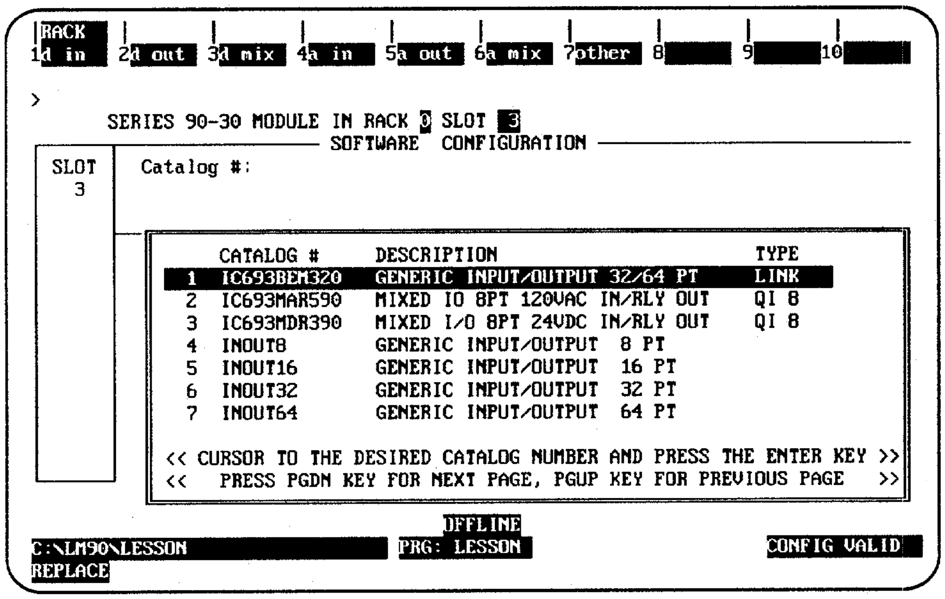
Table 10-2. Generic I/O Module Configuration

Generic Catalog Number	Generic Description	Function Key
INPUT8 INPUT16 INPUT32 INPUT64	Generic Input 8 PT Generic Input 16 PT Generic Input 32 PT Generic Input 64 PT	F1 (d in)
OUTPUT5/8 OUTPUT12/26 OUTPUT32 OUTPUT64	Generic Output 5/8 PT Generic Output 12/16 PT Generic Output 32 PT Generic Output 64 PT	F2 (d out)
ALGIN4	Generic Input Analog 4 CH	F4 (a in)
ALGOUT2 ALGOUT4	Generic Output Analog 2 CH Generic Output Analog 4 CH	F5 (a out)
INOUT8 INOUT16 INOUT32 INOUT64	Generic Input/Output 8 PT Generic Input/Output 16 PT Generic Input/Output 32 PT Generic Input/Output 64 PT	F3 (d mix)

Configuring an I/O Link Interface Module

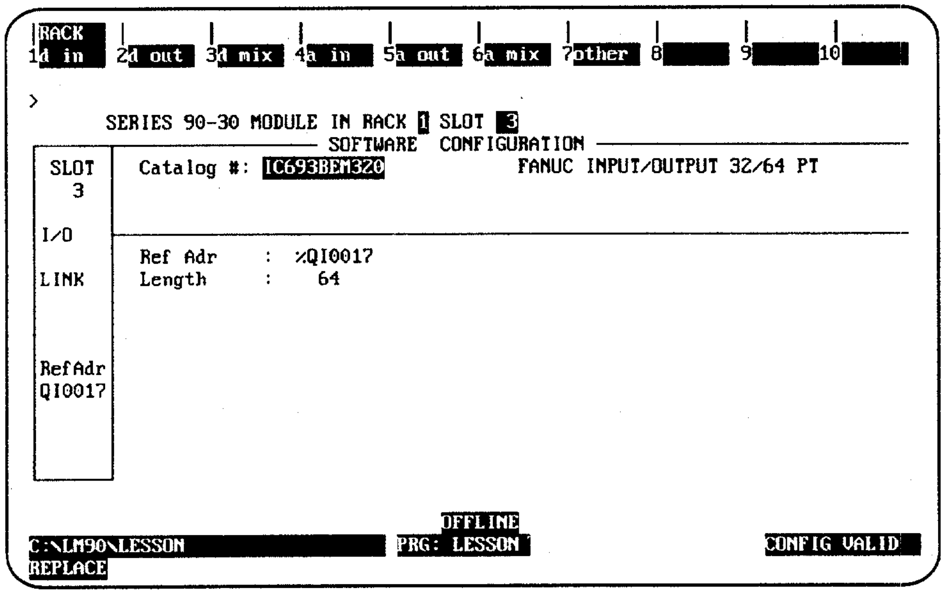
The Fanuc I/O link modules are grouped under the discrete mixed modules. To configure a Fanuc I/O link module on the I/O Configuration Rack screen:

1. Move the cursor to the desired rack and slot location. The slot may be either unconfigured or previously configured.
2. Press Module 30 I/O (**F1**) and then Discrete Mixed (**F3**) from the I/O Configuration Rack screen to display a list of catalog numbers and discrete mixed modules.



```
|RACK  |       |       |       |       |       |       |       |       |
1d in  2d out  3d mix  4a in   5a out  6a mix  7other  8       9       10
>
        SERIES 90-30 MODULE IN RACK 0 SLOT 3
                        SOFTWARE  CONFIGURATION
 SLOT   Catalog #:
  3

          CATALOG #      DESCRIPTION                        TYPE
        1  IC693BEM320   GENERIC INPUT/OUTPUT 32/64 PT      LINK
        2  IC693MAR590   MIXED IO 8PT 120VAC IN/RLY OUT     QI 8
        3  IC693MDR390   MIXED I/O 8PT 24VDC IN/RLY OUT     QI 8
        4  INOUT8        GENERIC INPUT/OUTPUT   8 PT
        5  INOUT16       GENERIC INPUT/OUTPUT  16 PT
        6  INOUT32       GENERIC INPUT/OUTPUT  32 PT
        7  INOUT64       GENERIC INPUT/OUTPUT  64 PT

     << CURSOR TO THE DESIRED CATALOG NUMBER AND PRESS THE ENTER KEY >>
     <<    PRESS PGDN KEY FOR NEXT PAGE, PGUP KEY FOR PREVIOUS PAGE   >>

                                 OFFLINE
C:\LM90\LESSON                   PRG: LESSON                CONFIG VALID
REPLACE
```

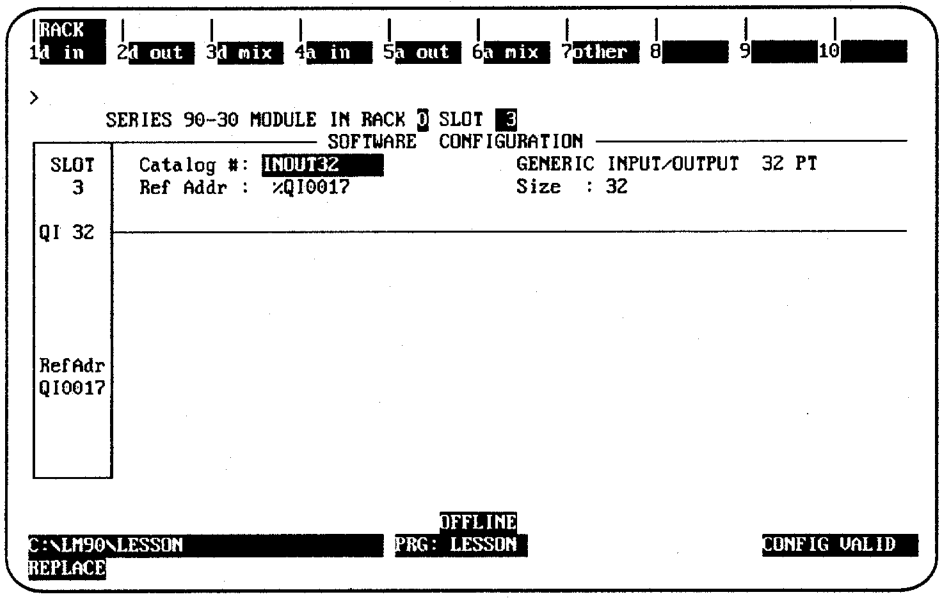
3. Position the cursor on the catalog number, IC693BEM320, and press the **Enter** key. The following detail screen is displayed.



```
|RACK  |       |       |       |       |       |       |       |       |
1d in  2d out  3d mix  4a in   5a out  6a mix  7other  8       9       10
>
        SERIES 90-30 MODULE IN RACK 0 SLOT 3
                        SOFTWARE  CONFIGURATION
 SLOT   Catalog #: IC693BEM320          FANUC INPUT/OUTPUT 32/64 PT
  3

 I/O
        Ref Adr    :  %QI0017
 LINK   Length     :    64


 RefAdr
 QI0017

                                 OFFLINE
C:\LM90\LESSON                   PRG: LESSON                CONFIG VALID
REPLACE
```

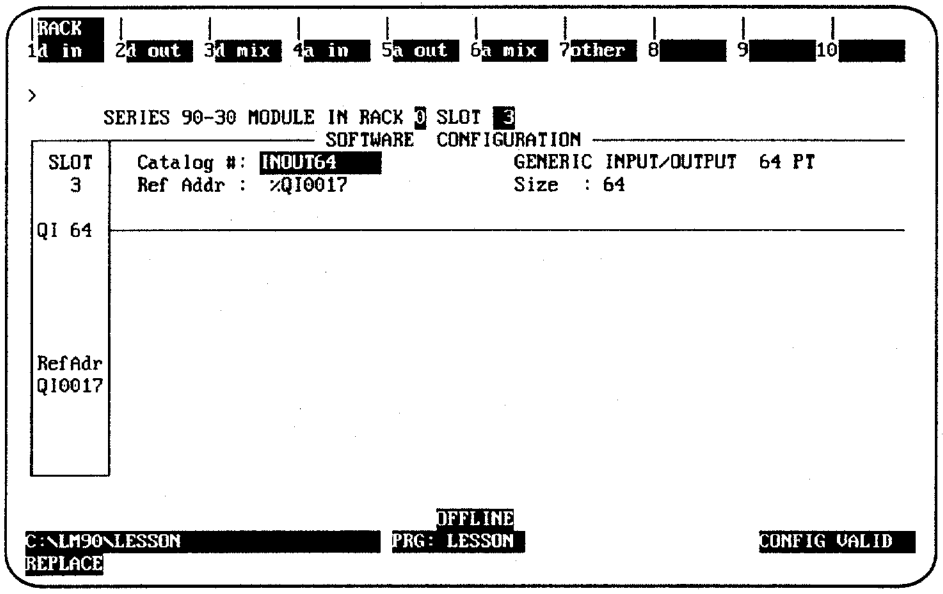
On this screen, the choices for the I/O address size are 32 and 64. The default value is 64.

4. If you select 32 as the reference size, the module will occupy 32 discrete inputs and 32 discrete output points. This is the same as if you select the catalog number INOUT32 for the Generic Input/Output 32-Point Module.



```
|RACK  |      |      |      |      |      |      |      |      |
1d in  2d out 3d mix 4a in  5a out 6a mix 7other 8      9      10

>
       SERIES 90-30 MODULE IN RACK 0 SLOT 3
                            SOFTWARE  CONFIGURATION
 SLOT  | Catalog #: INOUT32             GENERIC INPUT/OUTPUT  32 PT
  3    | Ref Addr :  %QI0017            Size  : 32
QI 32  |
RefAdr |
QI0017 |

                                      OFFLINE
C:\LM90\LESSON                        PRG: LESSON             CONFIG VALID
REPLACE
```

5. If you select 64 as the reference size, the module will occupy 64 discrete inputs and 64 discrete output points. This is the same as if you select the catalog number INOUT64 for the Generic Input/Output 64-Point Module.



```
|RACK  |      |      |      |      |      |      |      |      |
1d in  2d out 3d mix 4a in  5a out 6a mix 7other 8      9      10

>
       SERIES 90-30 MODULE IN RACK 0 SLOT 3
                            SOFTWARE  CONFIGURATION
 SLOT  | Catalog #: INOUT64             GENERIC INPUT/OUTPUT  64 PT
  3    | Ref Addr :  %QI0017            Size  : 64
QI 64  |
RefAdr |
QI0017 |

                                      OFFLINE
C:\LM90\LESSON                        PRG: LESSON             CONFIG VALID
REPLACE
```

Section 8: Configuring an HSC or Embedded HSC

The Series 90-30 High Speed Counter module may be configured to count either up or down, to count both up and down, or to count the difference between two changing values. The module is configured for the application to function in one of three ways:

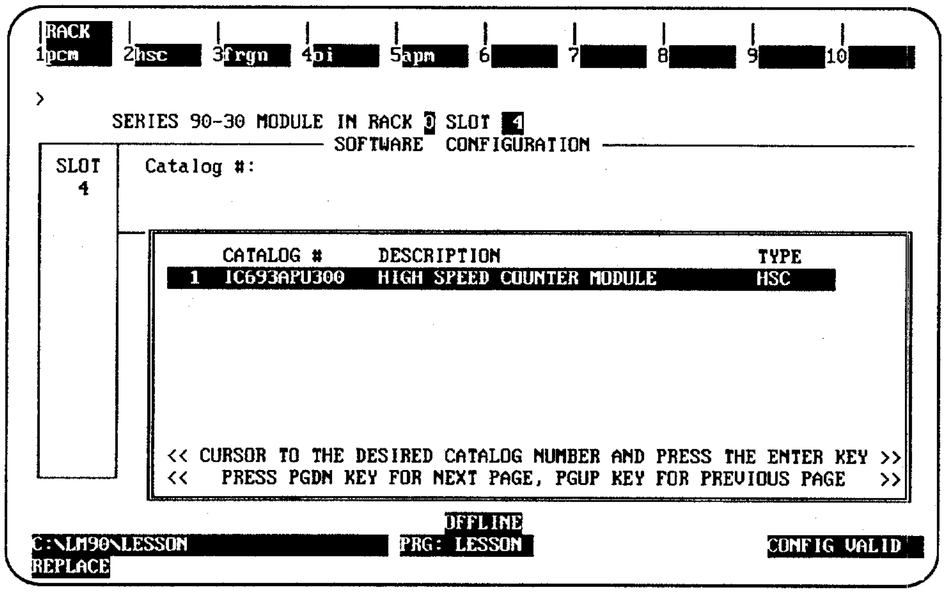
- As four 16-bit counters. Each of the four counters may independently count either up or down. This module configuration is referred to as "Type A." Each Type A counter has 3 inputs and 1 output.
- As two 32-bit counters. Each may independently operate in **UP/DOWN**, **PULSE/DIRECTION**, or **A QUAD B** mode. This configuration is referred to as "Type B." Each Type B counter has 6 inputs and 2 outputs.
- As one 32-bit differential counter, which can operate in **UP/DOWN**, **PULSE/DIRECTION**, or **A QUAD B** mode. This configuration is suitable for applications requiring motion control, differential counting, or homing capability. This counter uses all 12 of the module's inputs and all 4 outputs.

For more information on the High Speed Counter module, refer to GFK-0293.

The High Speed Counter module accepts 12 configurable input signals, and provides 4 output signals for counter operations. Many additional features can be changed in the configuration software for the application program.

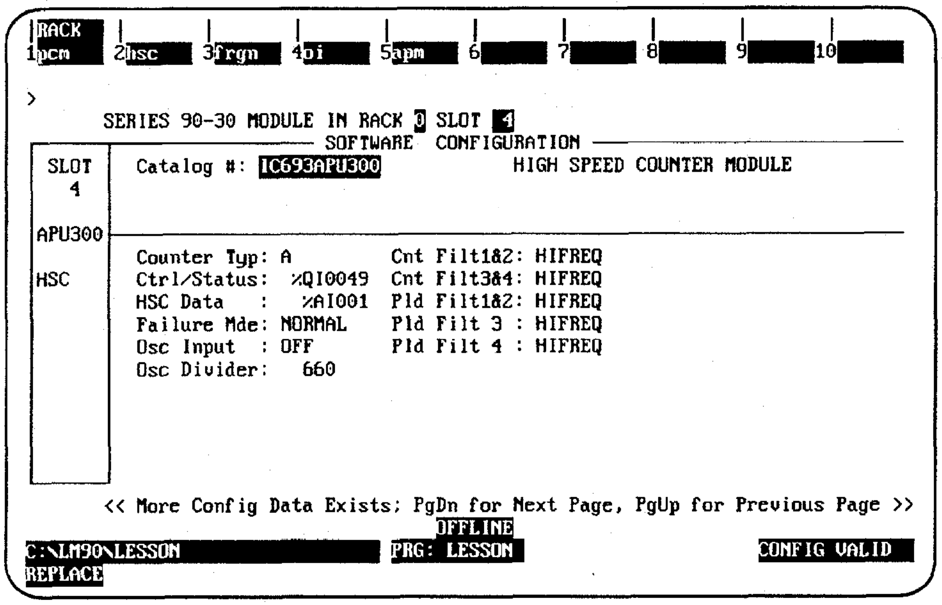
To configure the High Speed Counter module:

1. Move the cursor to the slot where the module will be located.
2. Press **Other** `(F8)` and then **High Speed Counter** `(F2)` to locate a High Speed Counter Module in the slot.



```
|RACK   |       |       |       |       |       |       |       |       |
1pcm    2hsc    3frgn   4oi     5apm    6       7       8       9       10

>
        SERIES 90-30 MODULE IN RACK 0 SLOT 4
                        SOFTWARE  CONFIGURATION
 SLOT   Catalog #:
  4

            CATALOG #      DESCRIPTION                       TYPE
         1  IC693APU300    HIGH SPEED COUNTER MODULE         HSC

     << CURSOR TO THE DESIRED CATALOG NUMBER AND PRESS THE ENTER KEY >>
     <<   PRESS PGDN KEY FOR NEXT PAGE, PGUP KEY FOR PREVIOUS PAGE   >>
                                  OFFLINE
C:\LM90\LESSON                  PRG: LESSON                 CONFIG VALID
REPLACE
```

3. Press the **Enter** key to enter the catalog number shown in reverse video. After pressing the **Enter** key, the following screen is displayed. This screen displays the module configuration parameters for a Type A counter.

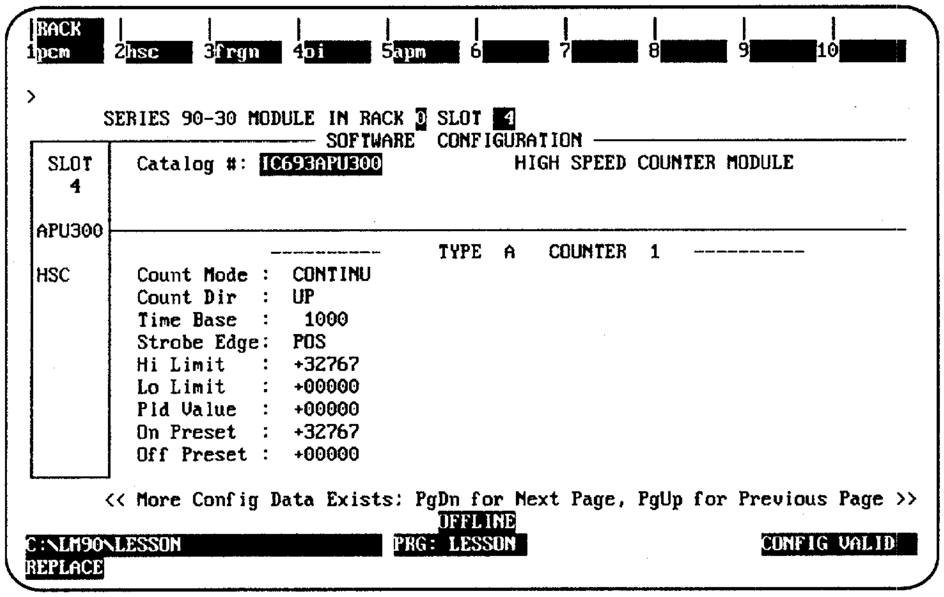


```
 RACK |      |       |      |       |       |       |       |       |
1 pcm  2 hsc  3 frgn  4 bi   5 apm   6       7       8       9       10
>
       SERIES 90-30 MODULE IN RACK 0 SLOT 4
                      SOFTWARE  CONFIGURATION
 SLOT    Catalog #: IC693APU300           HIGH SPEED COUNTER MODULE
  4

 APU300
         Counter Typ: A            Cnt Filt1&2: HIFREQ
 HSC     Ctrl/Status:  %QI0049     Cnt Filt3&4: HIFREQ
         HSC Data   :   %AI001     Pld Filt1&2: HIFREQ
         Failure Mde: NORMAL       Pld Filt 3 : HIFREQ
         Osc Input  : OFF          Pld Filt 4 : HIFREQ
         Osc Divider:   660

       << More Config Data Exists; PgDn for Next Page, PgUp for Previous Page >>
                                            OFFLINE
C:\LM90\LESSON                         PRG: LESSON                  CONFIG VALID
REPLACE
```

Parameter	Description
Counter Type	Select the desired counter type by entering **A***, **B**, or **C**; or,press the **Tab** key.
Control/Status	Select the %QI reference location for discrete input data that the High Speed Counter Module sends to the PLC, and discrete output data that the PLC sends to the HSC Module. The length of the discrete data is fixed at 16 bits (2 bytes). Discrete input data consists of status information, such as output status, strobe input status, preload input status, home input status (for type C), and module ready status. Discrete output data consists of control information, such as output enable/ disable, strobe enable/disable, counter enable/disable, preload enable/disable, clear error, and home command (for type C).
HSC Data	Select the %AI reference location for data sent to the PLC by the HSC module. The length of the data is fixed at 15 words. HSC Data consists of accumulated counts, strobe register contents, and the counter per timebase value.
Failure Mode	Enter a value to select an output failure mode. Choices are **NORMAL***, **FORCEOFF**, and **HOLDLAST**
Oscillator Input	This field allows the use of an internal oscillator reference in place of one count input. The base value is 660 Khz. The reference is divisible by a 16-bit value (refer to the Oscillator Divider explained below). Choices are **ON** or **OFF***. **ON** selects the input to be used as an oscillator reference; **OFF** selects the input to be used as a counter input.
Oscillator Divider	Select the 16-bit value that is divided into 660 Khz in order to achieve the final reference frequency. This value is only used if the ***Oscillator Input*** field above is set to ON.
Count Filter	This field identifies the Counter Input Filter.
Preload Filter	This field identifies the Counter Preload Filter.

* Default selection.

4. Press the **Page Down** key to display the detail screen for Type A Counter 1. Repeatedly press the **Page Down** key to scroll through the detail screen for each counter and then return to the main detail screen for a Type A counter. Press the **Page Up** key to display the detail screen for Type A Counter 4 and then scroll through the counters in reverse order.



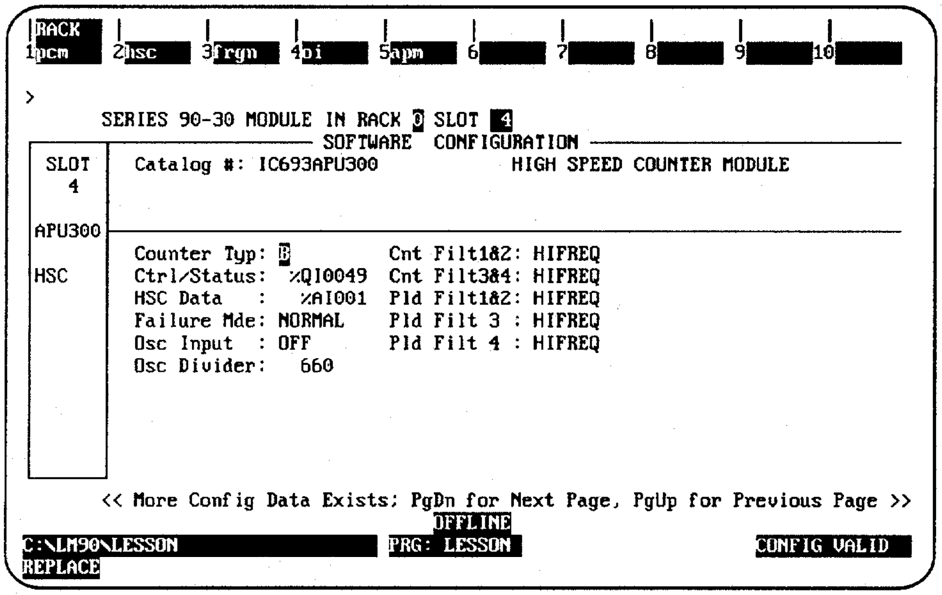
```
RACK
1pcm   2hsc   3frgn   4di   5apm   6   7   8   9   10

>
      SERIES 90-30 MODULE IN RACK 0 SLOT 4
                          SOFTWARE  CONFIGURATION
 SLOT    Catalog #: IC693APU300            HIGH SPEED COUNTER MODULE
  4

APU300
                    ----------    TYPE  A   COUNTER  1   ----------
HSC      Count Mode :  CONTINU
         Count Dir  :  UP
         Time Base  :   1000
         Strobe Edge:  POS
         Hi Limit   :  +32767
         Lo Limit   :  +00000
         Pld Value  :  +00000
         On Preset  :  +32767
         Off Preset :  +00000

      << More Config Data Exists: PgDn for Next Page, PgUp for Previous Page >>
                                    OFFLINE
C:\LM90\LESSON                    PRG: LESSON                    CONFIG VALID
REPLACE
```

Parameter	Description
Count Mode	Select continuous (**CONTINU***) or single-shot (**SINGSHOT**) counting.
Count Direction	Select the count direction for the type A counter. Choices are **UP*** or **DOWN**.
Time Base	Select the counts per time base (**0** to **+65,535**). (Default = **1000**)
Strobe Edge	Specify the positive (**POS***) or negative (**NEG**) edge active for the strobe input.
High Limit	Select the highest value for the count accumulator (**-32,768** to **+32,767**). (Default = **32,767**)
Low Limit	Select the lowest value for the count accumulator (**-32,768** to **+32,767**). (Default = **0**)
Preload Value	Select the accumulator preload value (**-32,768** to **+32,767**). (Default = **0**)
On Preset	Select the output on accumulator value (**-32,768** to **+32,767***).
Off Preset	Select the output off accumulator value (**-32,768** to **+32,767**). (Default = **0**)

* Default selection.

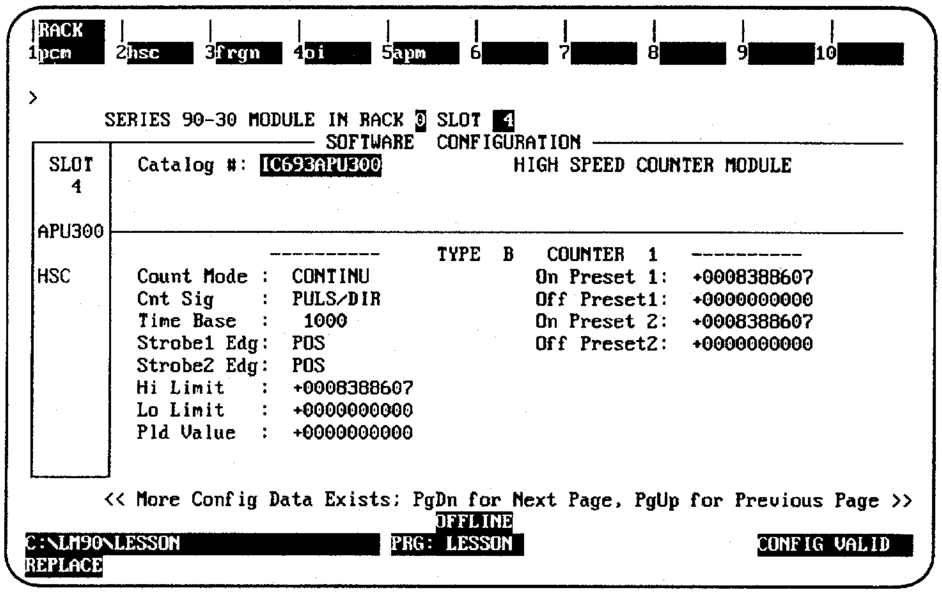
5. To change from a Type A to Type B counter, press the **Tab** key with the cursor positioned on the *Counter Type* field on the main detail screen for the Type A counter. When B is displayed in this field, press the **Enter** key.

   This example screen lists the module configuration parameters for a Type B counter.



```
|RACK  |      |      |      |      |      |      |      |      |      |
1pcm   2hsc   3frgn  4bi    5apm   6      7      8      9      10

>
      SERIES 90-30 MODULE IN RACK 0 SLOT 4
 ----------------------- SOFTWARE  CONFIGURATION --------------------
 SLOT | Catalog #: IC693APU300          HIGH SPEED COUNTER MODULE
  4   |
      |
APU300|----------------------------------------------------------------
      | Counter Typ: B           Cnt Filt1&2: HIFREQ
HSC   | Ctrl/Status:  %Q10049    Cnt Filt3&4: HIFREQ
      | HSC Data   :   %AI001    Pld Filt1&2: HIFREQ
      | Failure Mde: NORMAL      Pld Filt 3 : HIFREQ
      | Osc Input  : OFF         Pld Filt 4 : HIFREQ
      | Osc Divider:   660

      << More Config Data Exists; PgDn for Next Page, PgUp for Previous Page >>
                                                 OFFLINE
C:\LM90\LESSON                            PRG: LESSON              CONFIG VALID
REPLACE
```

6. Press the **Page Down** key to display the detail screen for Type B Counter 1. Repeatedly press the **Page Down** or **Page Up** key to scroll through the detail screen for each counter and then return to the main detail screen for the Type B counter.



```
|RACK  |      |      |      |      |      |      |      |      |      |
1pcm   2hsc   3frgn  4bi    5apm   6      7      8      9      10

>
      SERIES 90-30 MODULE IN RACK 0 SLOT 4
 ----------------------- SOFTWARE  CONFIGURATION --------------------
 SLOT | Catalog #: IC693APU300          HIGH SPEED COUNTER MODULE
  4   |
      |
APU300|----------------------------------------------------------------
      |             ----------     TYPE  B    COUNTER  1    ----------
HSC   | Count Mode  :  CONTINU               On Preset 1:  +0008388607
      | Cnt Sig     :  PULS/DIR              Off Preset1:  +0000000000
      | Time Base   :   1000                 On Preset 2:  +0008388607
      | Strobe1 Edg :  POS                   Off Preset2:  +0000000000
      | Strobe2 Edg :  POS
      | Hi Limit    :  +0008388607
      | Lo Limit    :  +0000000000
      | Pld Value   :  +0000000000

      << More Config Data Exists; PgDn for Next Page, PgUp for Previous Page >>
                                                 OFFLINE
C:\LM90\LESSON                            PRG: LESSON              CONFIG VALID
REPLACE
```

Note

Because there are two 32-bit counters, there will be two screens similar to these example screens.

Parameter	Description
Count Mode	Select continuous (**CONTINU***) or single-shot (**SINGSHOT**) counting.
Count Signal	For Type B or C counter, select how each counter's signals will be used. Choices are **UP/DN**, **PULS/DIR***, or **AQUADB**.
Time Base	Select the counts per time base (**0** to **+65,535**). (Default = **1000**)
Strobe Edge	Specify the positive or negative edge active for the strobe input. Choices are **POS*** or **NEG**.
High Limit	Select the highest value for the count accumulator (**-2,147,483,648** to **+2,147,483,647**). (Default = **+8,388,607**)
Low Limit	Select the lowest value for the count accumulator (**-2,147,483,648** to **+2,147,483,647**). (Default = **0**)
Preload Value	Select the accumulator preload value (**-2,147,483,648** to **+2,147,483,647**). (Default = **0**)
On Preset	Select the output on accumulator value (**-2,147,483,648** to **+2,147,483,647**). (Default = **+8,388,607**)
Off Preset	Select the output off accumulator value (**-2,147,483,648** to **+2,147,483,647**). (Default = **0**)

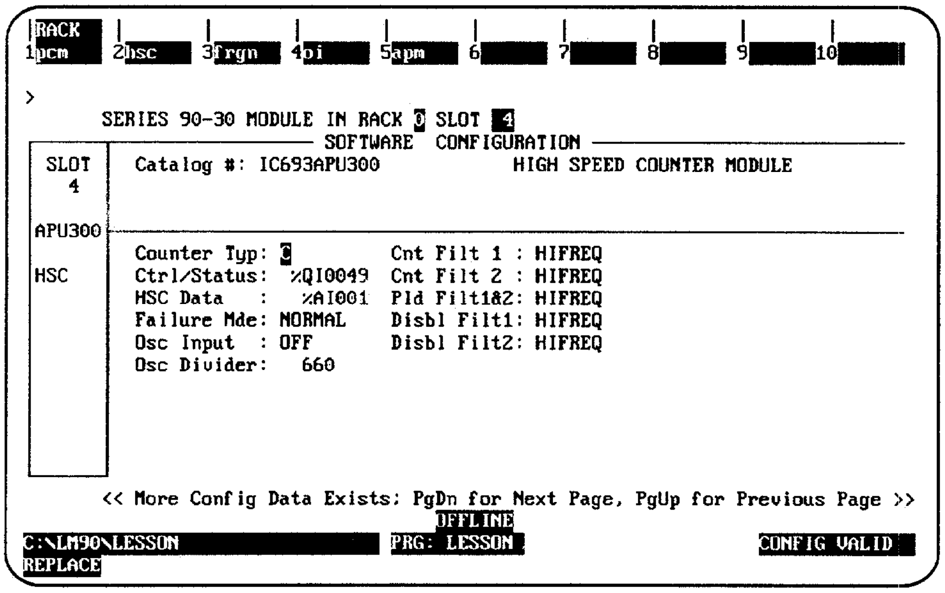
* Default selection.

Note

The Low Limit and High Limit parameters must be validated so that low limit is less than or equal to the high limit.

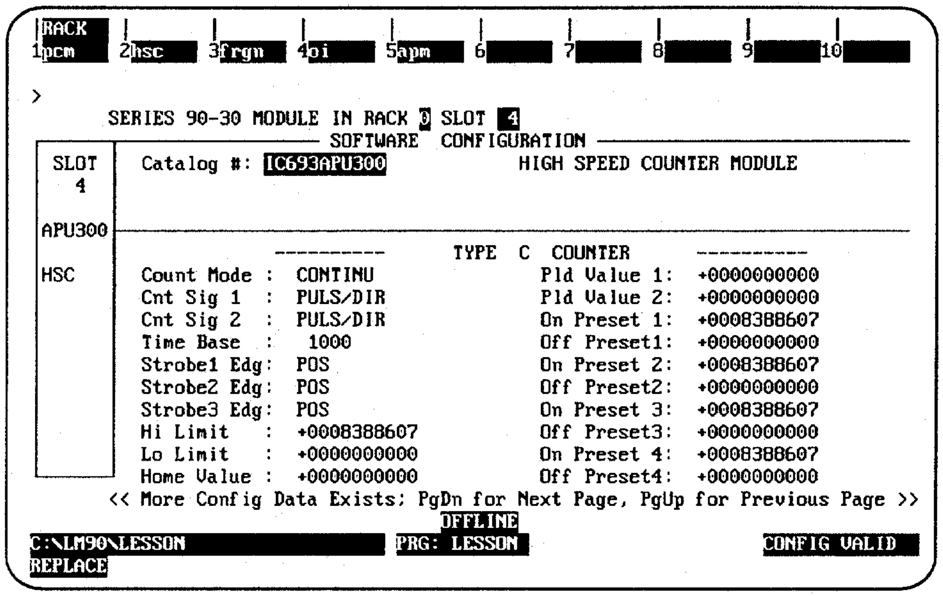
7. To change from a Type B to Type C counter, press the **Tab** key with the cursor positioned on the *Counter Type* field on the main detail screen for the Type B counter. When C is displayed in this field, press the **Enter** key.

   This example screen lists the module configuration parameters for a Type C counter.



```
RACK |    |    |    |    |    |    |    |    |    |
1pcm  2hsc  3rgn  4oi  5apm  6    7    8    9    10

>
         SERIES 90-30 MODULE IN RACK 0 SLOT 4
                    ------------ SOFTWARE CONFIGURATION ------------
 SLOT  | Catalog #: IC693APU300          HIGH SPEED COUNTER MODULE
  4    |
       |
APU300 |------------------------------------------------------------
       | Counter Typ: C          Cnt Filt 1 : HIFREQ
HSC    | Ctrl/Status:  %QI0049   Cnt Filt 2 : HIFREQ
       | HSC Data   :  %AI001    Pld Filt1&2: HIFREQ
       | Failure Mde: NORMAL     Disbl Filt1: HIFREQ
       | Osc Input  : OFF        Disbl Filt2: HIFREQ
       | Osc Divider:   660

    << More Config Data Exists: PgDn for Next Page, PgUp for Previous Page >>
                                  OFFLINE
C:\LM90\LESSON                    PRG: LESSON               CONFIG VALID
REPLACE
```

8. Press the **Page Down** key to display the detail screen for a Type C counter. Press the **Page Down** key to return to the main detail screen for the module.



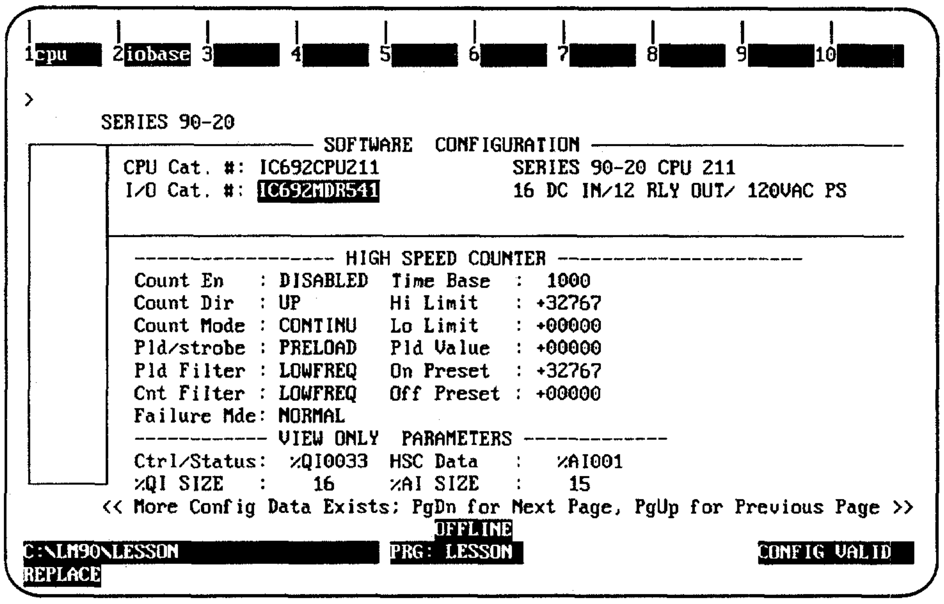
```
RACK
1pcm     2hsc     3frgn    4oi      5apm     6        7        8        9        10

>
      SERIES 90-30 MODULE IN RACK 0 SLOT 4
                          SOFTWARE  CONFIGURATION
 SLOT    Catalog #: IC693APU300             HIGH SPEED COUNTER MODULE
  4

APU300
              ----------       TYPE  C  COUNTER        ----------
HSC      Count Mode :  CONTINU               Pld Value 1:  +0000000000
         Cnt Sig 1  :  PULS/DIR              Pld Value 2:  +0000000000
         Cnt Sig 2  :  PULS/DIR              On Preset 1:  +0008388607
         Time Base  :   1000                 Off Preset1:  +0000000000
         Strobe1 Edg:  POS                   On Preset 2:  +0008388607
         Strobe2 Edg:  POS                   Off Preset2:  +0000000000
         Strobe3 Edg:  POS                   On Preset 3:  +0008388607
         Hi Limit   :  +0008388607           Off Preset3:  +0000000000
         Lo Limit   :  +0000000000           On Preset 4:  +0008388607
         Home Value :  +0000000000           Off Preset4:  +0000000000
      << More Config Data Exists: PgDn for Next Page, PgUp for Previous Page >>
                                       OFFLINE
C:\LM90\LESSON                    PRG: LESSON                    CONFIG VALID
REPLACE
```

The configurable parameters on this screen are as described previously for the Type B counter.

Use these detail screens to configure the features of the High Speed Counter module.

Configuring the Embedded HSC for a CPU 211

The High Speed Counter module for a CPU 211 can only be a Type A counter.



```
 |      |       |      |      |      |      |      |      |      |
1cpu   2iobase 3      4      5      6      7      8      9      10
>
        SERIES 90-20
                         SOFTWARE  CONFIGURATION
          CPU Cat. #:  IC692CPU211             SERIES 90-20 CPU 211
          I/O Cat. #:  IC692MDR541             16 DC IN/12 RLY OUT/ 120VAC PS

           ------------------ HIGH SPEED COUNTER ----------------------
           Count En    : DISABLED  Time Base   :  1000
           Count Dir   : UP        Hi Limit    : +32767
           Count Mode  : CONTINU   Lo Limit    : +00000
           Pld/strobe  : PRELOAD   Pld Value   : +00000
           Pld Filter  : LOWFREQ   On Preset   : +32767
           Cnt Filter  : LOWFREQ   Off Preset  : +00000
           Failure Mde: NORMAL
           ------------ VIEW ONLY  PARAMETERS -------------
           Ctrl/Status:  %QI0033   HSC Data    :   %AI001
           %QI SIZE    :    16     %AI SIZE    :    15
        << More Config Data Exists: PgDn for Next Page, PgUp for Previous Page >>
                                             OFFLINE
C:\LM90\LESSON                         PRG: LESSON                      CONFIG VALID
REPLACE
```

The HSC module may be configured after paging down from the CPU 211 detail screen. Additional HSC parameters may be configured, as described in the following table.

Parameter	Description
Count Enable	Enable or disable the counter. Choices are **ENABLE** or **DISABLE***.
Count Direction	Select the count direction for the module. Choices are **UP*** or **DOWN**.
Count Mode	Select continuous (**CONTINU***) or single-shot (**SINGSHOT**) counting.
Preload/Strobe	Select **PRELOAD*** or **STROBE SELECT**
Preload Filter	Select **HIFREQ** or **LOFREQ***.
Count Filter	Select **HIFREQ** or **LOFREQ***.
Failure Mode	Choices are **HOLDLAST**, **NORMAL***, or **FORCEOFF**.
Time Base	Select the counts per time base (**0** to **+65,535**). (Default = **1000**)
High Limit	Select the highest value for the count accumulator (**-32,768** to **+32,767**). (Default = **+32,767**)
Low Limit	Select the lowest value for the count accumulator (**-32,768** to **+32,767**). (Default = **0**)
Preload Value	Select the accumulator preload value (**-32,768** to **+32,767**). (Default = **0**)
On Preset	Select the output on accumulator value (**-32,768** to **+32,767**). (Default = **+32,767**)
Off Preset	Select the output off accumulator value (**-32,768** to **+32,767**). (Default = **0**)

* Default selection.

Section 9: Configuring a PCM Module

The Programmable Coprocessor Module (PCM) combines the function of a Communications Module and the ASCII/BASIC Module into a single module. It may be configured to behave as one CCM port, two independent CCM ports, one CCM port and one BASIC application having one port, or one BASIC application using one or both serial ports.

The following modes of configuration are available:

Mode	Description
PCM CFG	All configuration data for the PCM is located in the User Configuration Data File (UCDF), created and loaded to the PCM using PCOP.
PROG PRT	Select Programmer Port mode.
CCM ONLY	Select CCM on both ports 1 and 2.
PROG/CCM	Select port 1 for programmer connection and port 2 for CCM.
CCM/PROG	Select port 1 for CCM and port 2 for the programmer.
BASIC	Select both ports 1 and 2 for BASIC.
BAS/CCM	Select port 1 for BASIC configuration and port 2 for CCM.

Logicmaster 90-30/20/Micro programming software provides configuration support of the following modules:

- PCM with 160 KB memory module.
- PCM (multiple mode) with 192 KB memory module.
- PCM (multiple mode) with 640 KB memory module.

Available modes for each module are listed below:

Mode	Module Type		
	160K PCM	192K PCM	640K PCM
PCM CFG	Yes	Yes	Yes
PROG PRT	Yes	Yes	Yes
CCM ONLY	Yes *	Yes	Yes
PROG/CCM	Yes *	Yes	Yes
CCM/PROG	Yes *	Yes	Yes
BASIC	Yes	Yes	Yes
BAS/CCM	Yes	Yes	Yes

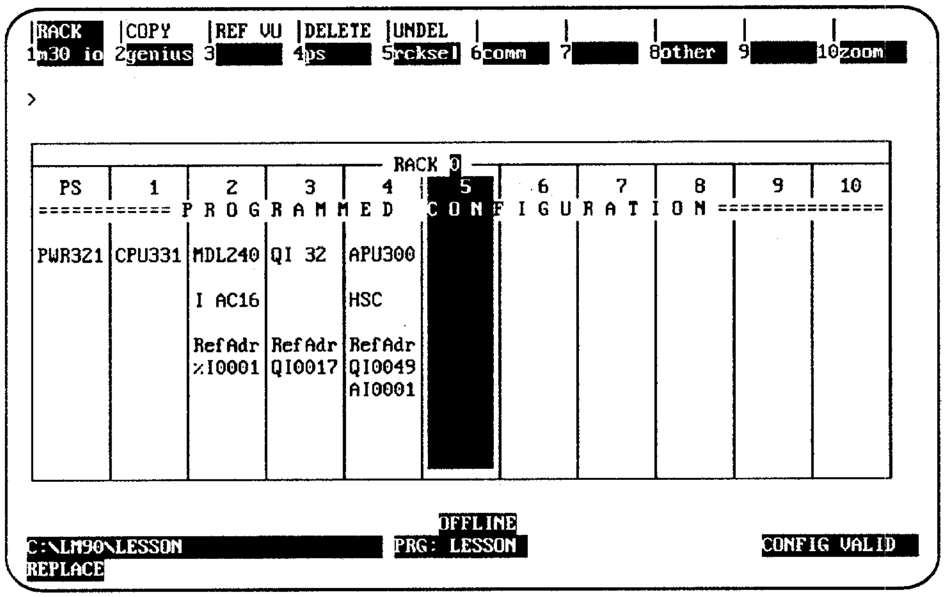
* The interface for port 2 is fixed at RS-485 for the 160K modules.

For more information on the PCM, refer to the *Series 90 Programmable Coprocessor Module and Support Software User's Manual*, GFK-0255.

Configuring a PCM

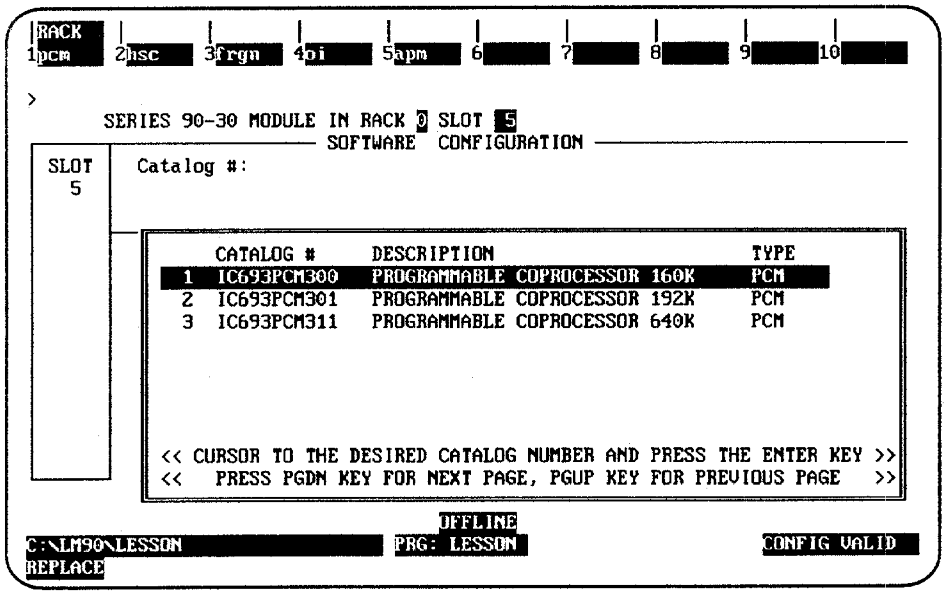
To configure a PCM on the I/O Configuration Rack screen:

1. Move the cursor to the desired rack and slot location. The slot may be either unconfigured or previously configured.



```
 RACK    |COPY    |REF VU  |DELETE  |UNDEL   |        |        |        |        |
1m30 io 2genius  3        4ps      5rcksel  6comm    7        8other   9       10zoom

>

                                      RACK 0
  PS     |   1   |   2   |   3   |   4   |   5   |   6   |   7   |   8   |   9   |  10
=========== P R O G R A M M E D   C O N F I G U R A T I O N ==============

PWR321 |CPU331 |MDL240 |QI 32  |APU300 |
       |       |I AC16 |       |HSC    |
       |       |RefAdr |RefAdr |RefAdr |
       |       |%I0001 |QI0017 |QI0049 |
       |       |       |       |AI0001 |

                                              OFFLINE
C:\LM90\LESSON                              PRG: LESSON                  CONFIG VALID
REPLACE
```

2. Press **Other** **(F8)** and then **PCM** **(F1)** from the I/O Configuration Rack screen to display a list of catalog numbers and modules.



```
 RACK    |        |        |        |        |        |        |        |        |
1pcm    2hsc     3frgn    4oi      5apm     6        7        8        9       10

>
       SERIES 90-30 MODULE IN RACK 0 SLOT 5
                              SOFTWARE  CONFIGURATION
 SLOT    Catalog #:
  5

            CATALOG #      DESCRIPTION                          TYPE
        1   IC693PCM300    PROGRAMMABLE COPROCESSOR 160K        PCM
        2   IC693PCM301    PROGRAMMABLE COPROCESSOR 192K        PCM
        3   IC693PCM311    PROGRAMMABLE COPROCESSOR 640K        PCM

        << CURSOR TO THE DESIRED CATALOG NUMBER AND PRESS THE ENTER KEY >>
        <<   PRESS PGDN KEY FOR NEXT PAGE, PGUP KEY FOR PREVIOUS PAGE   >>

                                              OFFLINE
C:\LM90\LESSON                              PRG: LESSON                  CONFIG VALID
REPLACE
```

3. Position the cursor on the catalog number for the module you wish to select, and press the **Enter** key. The module detail screen for the module you selected will be displayed.

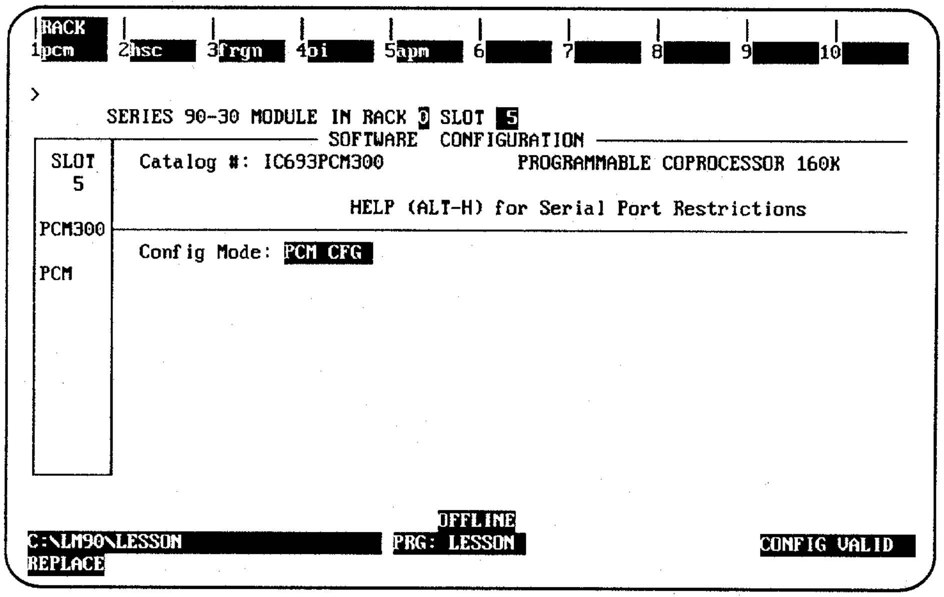
Selecting the Configuration Mode

The configuration mode is selected in the *Configuration Mode* field on the module detail screen. (The module detail screen is displayed by positioning the cursor on the catalog number for the module you wish to select and pressing the **Enter** key.)

1. To select a different configuration mode, move the cursor to the *Configuration Mode* field and repeatedly press the **Tab** key until the desired mode is displayed on the screen. Then, press the **Enter** key.
2. Complete the detail screen for the desired configuration mode, and press the **Enter** key. For help selecting parameters, press **ALT-H**.

PCM CFG Mode

When **PCM CFG** mode is selected, the following PCM detail screen is displayed.

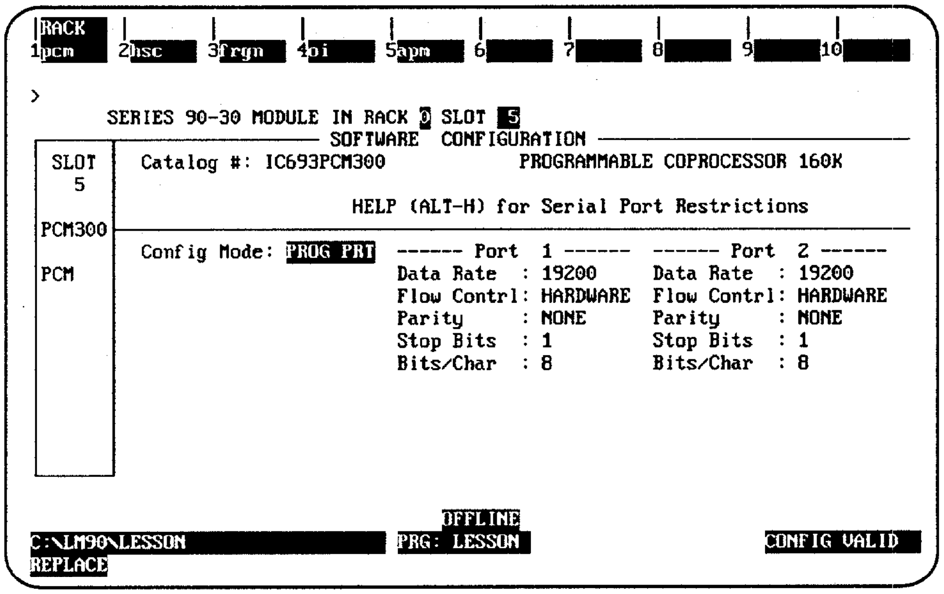


```
|RACK  |      |      |      |      |      |      |      |      |
1pcm   2hsc   3frgn  4oi    5apm   6      7      8      9      10
>
       SERIES 90-30 MODULE IN RACK 0 SLOT 5
 ------------------------------ SOFTWARE  CONFIGURATION ------------------
 SLOT  | Catalog #: IC693PCM300          PROGRAMMABLE COPROCESSOR 160K
   5   |
       |                    HELP (ALT-H) for Serial Port Restrictions
PCM300 |-----------------------------------------------------------------
       | Config Mode: PCM CFG
PCM    |

                                        OFFLINE
C:\LM90\LESSON                         PRG: LESSON              CONFIG VALID
REPLACE
```

In this screen, the configuration mode is set to **PCM CFG** for the PCM Configuration Data File mode. This mode uses the User Configuration Data (UCDF) that was loaded into a user module named UCDF in the PCMs battery-backed RAM memory. (The UCDF is created using PCOP software.) The configuration data is used for building the configuration of the PCM during the power-up sequence, initializing available hardware on the PCM and specifying the user or system tasks to be started. For more information, refer to the *Series 90 Programmable Coprocessor Module and Support Software User's Manual,* GFK-0255.

PROG PRT Mode

When **PROG PRT** (Programmer Port) mode is selected, the following PCM detail screen is displayed.



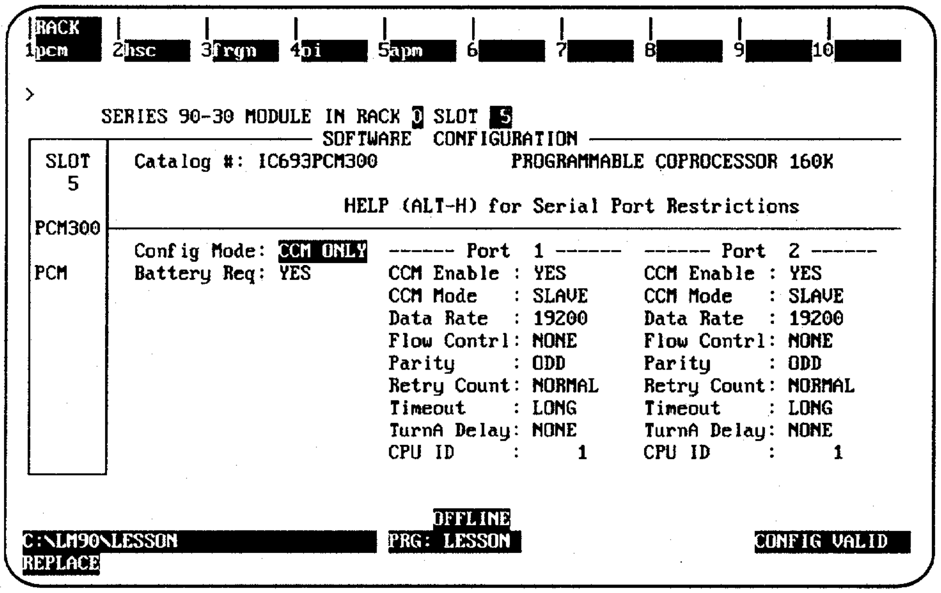
```
RACK
1pcm  2hsc  3rgn  4oi  5apm  6  7  8  9  10
>
     SERIES 90-30 MODULE IN RACK 0 SLOT 5
                       SOFTWARE  CONFIGURATION
 SLOT   Catalog #: IC693PCM300             PROGRAMMABLE COPROCESSOR 160K
  5
                           HELP (ALT-H) for Serial Port Restrictions
PCM300
        Config Mode: PROG PRT  ------ Port  1 ------   ------ Port  2 ------
PCM                            Data Rate  : 19200      Data Rate  : 19200
                               Flow Contrl: HARDWARE   Flow Contrl: HARDWARE
                               Parity     : NONE       Parity     : NONE
                               Stop Bits  : 1          Stop Bits  : 1
                               Bits/Char  : 8          Bits/Char  : 8

                                            OFFLINE
C:\LM90\LESSON                             PRG: LESSON              CONFIG VALID
REPLACE
```

Parameter	Description
Configuration Mode	The configuration mode is set to **PROG PRT**
Interface	The interface parameter for port 1 can only be **RS-232**; port 2 of the 192K and 640K modules may have an interface value of either **RS-232*** or **RS-485**. The interface parameter is not used or displayed for the 160K module.
Data Rate	Data rate (bits per second or bps) for the port. Choices are **300**, **600**, **1200**, **2400**, **4800**, **9600**, or **19200***.
Flow Control	Type of flow control to be used on the port. Choices are **HARDWARE***, **SOFTWARE**, or **NONE**.
Parity	Type of parity to be used on the port. Choices are **NONE***, **EVEN**, or **ODD**.
Stop Bits	Number of stop bits for the port. Choices are **1*** or **2**.
Bits per Character	Number of bits per character for data transfer on the port. Choices are **7** or **8***.

* Default selection.

CCM ONLY Mode

When **CCM ONLY** mode is selected, the following PCM detail screen is displayed.



```
 RACK
1pcm     2hsc     3frgn    4oi      5apm     6        7        8        9        10
>
        SERIES 90-30 MODULE IN RACK 0 SLOT 5
                          SOFTWARE  CONFIGURATION
 SLOT     Catalog #: IC693PCM300              PROGRAMMABLE COPROCESSOR 160K
  5
                          HELP (ALT-H) for Serial Port Restrictions
 PCM300
          Config Mode: CCM ONLY  ------ Port  1 ------     ------ Port  2 ------
 PCM      Battery Req: YES       CCM Enable : YES          CCM Enable : YES
                                 CCM Mode   : SLAVE        CCM Mode   : SLAVE
                                 Data Rate  : 19200        Data Rate  : 19200
                                 Flow Contrl: NONE         Flow Contrl: NONE
                                 Parity     : ODD          Parity     : ODD
                                 Retry Count: NORMAL       Retry Count: NORMAL
                                 Timeout    : LONG         Timeout    : LONG
                                 TurnA Delay: NONE         TurnA Delay: NONE
                                 CPU ID     :      1       CPU ID     :      1

                                              OFFLINE
C:\LM90\LESSON                               PRG: LESSON                   CONFIG VALID
REPLACE
```

Parameter	Description
Configuration Mode	The configuration mode is set to **CCM ONLY**.
Battery Required	Specify whether a battery is required. Choices are **YES*** or **NO**.
CCM Enable	Specify whether the port is to be configured for use as a CCM port. Choices are **YES*** or **NO**.
CCM Mode	This parameter displays the availability of ports for CCM access. Choices are **SLAVE***, **PEER**, or **MASTER**.
Interface	The interface parameter for port 1 can only be **RS-232**; port 2 of the 192K and 640K modules may have an interface value of either **RS-232*** or **RS-485**. The interface parameter is not used or displayed for the 160K module.
Data Rate	Data rate (bits per second or bps) for the port. Choices are **300**, **600**, **1200**, **2400**, **4800**, **9600**, or **19200***.
Flow Control	Type of flow control to be used on the port. Choices are **HARDWARE** or **NONE***.
Parity	Type of parity to be used on the port. Choices are **NONE**, **EVEN**, or **ODD***.
Retry Count	Retry counts for CCM mode. Choices are **NORMAL*** or **SHORT**
Timeout	Length of timeouts used for CCM on the port. Choices are **LONG***, **MEDIUM**, **SHORT**, or **NONE**.
Turnaround Delay	Turnaround delay time to be used for CCM on the port. Choices are **NONE***, **10 ms**, **100 ms**, or **500 ms**.
CPU ID	Address of the port on a multi-drop network. This value is used to calculate the backoff delay upon an inquiry collision in **PEER** mode. The range of values allowed in this field is **1*** to **254**.

* Default selection.

PROG/CCM Mode

PROG/CCM mode selects port 1 as the programmer port and port 2 for CCM. The parameters for port 1 are the same as for **PROG PRT** mode; and for port 2, the parameters are the same as for **CCM ONLY** mode.

CCM/PROG Mode

CCM/PROG mode selects port 1 for CCM and port 2 as the programmer port. The parameters for port 1 are the same as for **CCM ONLY** mode; and for port 2, the parameters are the same as for **PROG PRT** mode.

BASIC Mode

BASIC mode selects BASIC configuration for both ports 1 and 2. The parameters for ports 1 and 2 are the same as for **PROG PRT** mode.

BAS/CCM Mode

BAS/CCM mode selects BASIC configuration for port 1 and CCM for port 2. The parameters for port 1 are the same as for **PROG PRT** mode; and for port 2, the parameters are the same as for **CCM ONLY** mode.

Section 10: *Configuring a TCP/IP Ethernet Module*

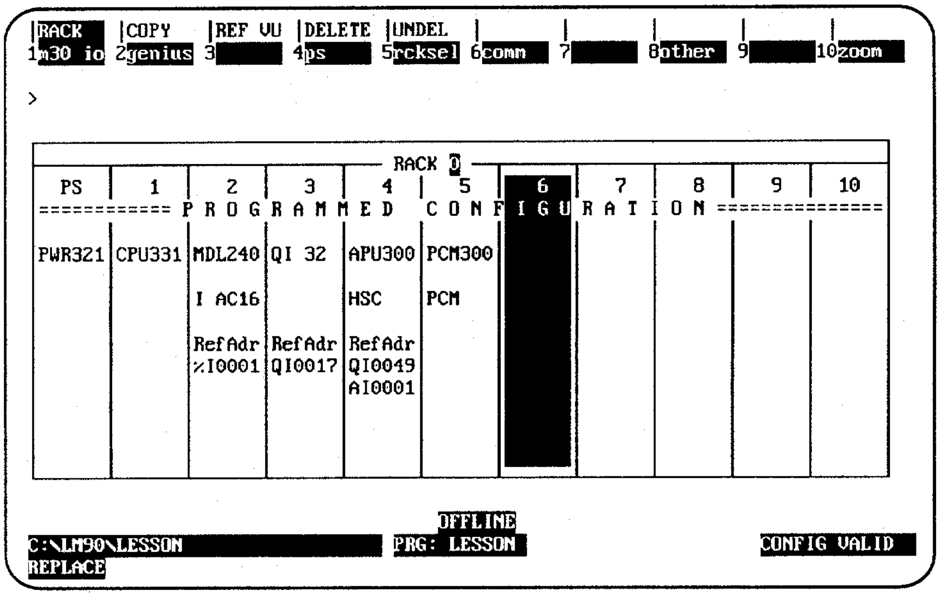
Logicmaster 90-30/20/Micro programming software provides configuration support of the TCP/IP Ethernet Module. For details about the TCP/IP Ethernet Module, refer to the *TCP/IP Ethernet Communications for the Series 90™-30 PLC* manual (GFK-1084).

Use the information in this section in conjunction with the "Procedure 2: Configuring the Ethernet Interface with Logicmaster 90-30 Configuration Software" section of chapter 2 of the *TCP/IP Ethernet Communications for the Series 90™-30 PLC* manual (GFK-1084).

Configuring a TCP/IP Ethernet Module

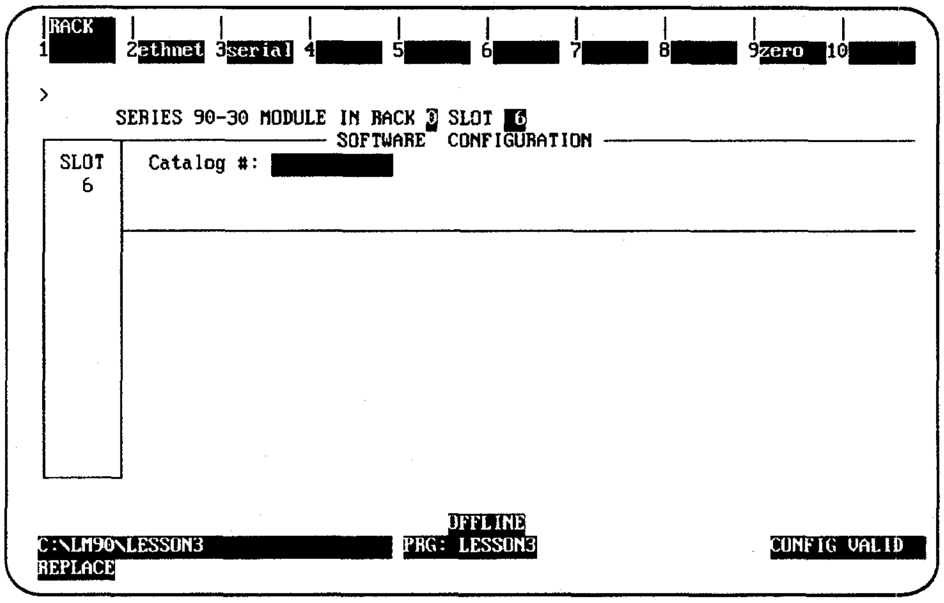
To configure a TCP/IP Ethernet Module on the I/O Configuration rack screen, do the following:

1. Move the cursor to the desired rack and slot location. The slot may be either unconfigured or previously configured.



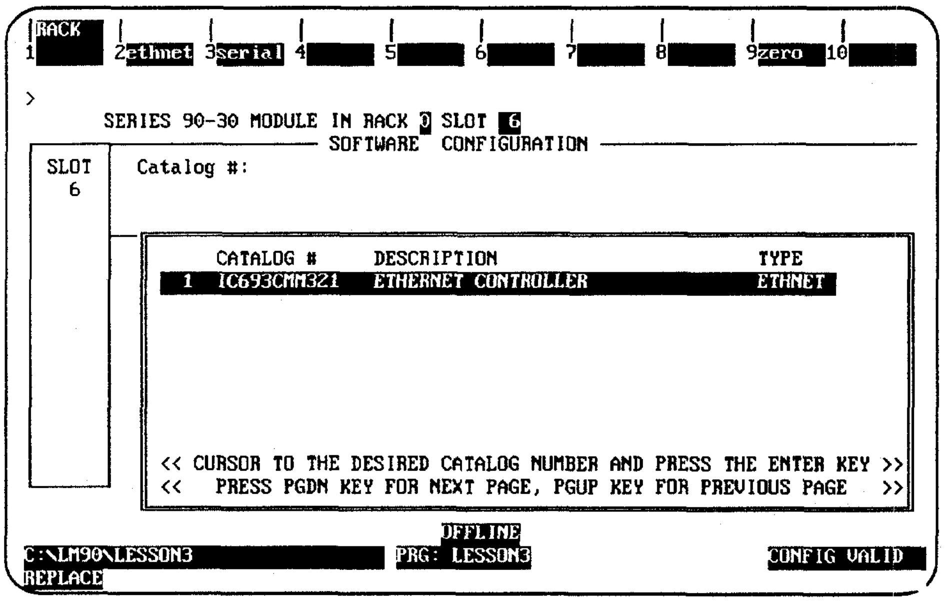
```
|RACK    |COPY     |REF VU  |DELETE  |UNDEL   |         |         |         |         |
1m30 io  2genius   3        4ps      5rcksel  6comm     7         8other    9         10zoom

>

                                      RACK 0
  PS   |  1   |  2   |  3   |  4   |  5   |  6   |  7   |  8   |  9   |  10
============ P R O G R A M M E D   C O N F I G U R A T I O N ==============

PWR321 |CPU331|MDL240|QI 32 |APU300|PCM300|      |      |      |      |
       |      |I AC16|      |HSC   |PCM   |      |      |      |      |
       |      |RefAdr|RefAdr|RefAdr|      |      |      |      |      |
       |      |%I0001|QI0017|QI0049|      |      |      |      |      |
       |      |      |      |AI0001|      |      |      |      |      |

                                        OFFLINE
C:\LM90\LESSON                        PRG: LESSON                 CONFIG VALID
REPLACE
```

2. Press the **Communications** softkey, i.e., **Comm (F6)**. Your screen display will change to the one shown on the following page.



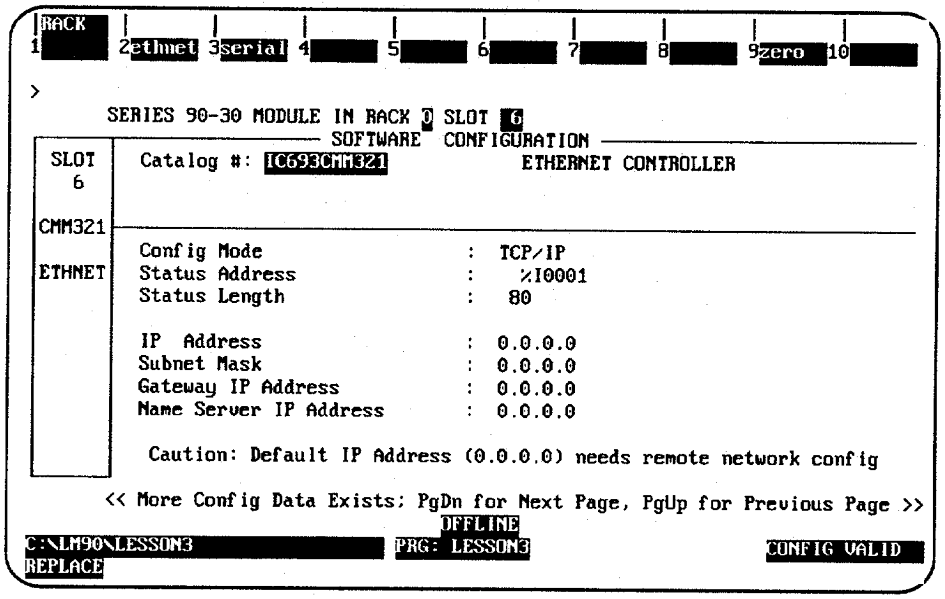
```
 RACK     |        |        |        |        |        |        |        |        |
1        2ethnet  3serial  4        5        6        7        8        9zero    10
>
        SERIES 90-30 MODULE IN RACK 0 SLOT 6
                               SOFTWARE  CONFIGURATION
 SLOT    Catalog #:
   6

C:\LM90\LESSON3                               OFFLINE
REPLACE                                      PRG: LESSON3                CONFIG VALID
```

3. Press **ethnet** (**F2**). Your screen display will change to the one shown below.



```
 RACK     |        |        |        |        |        |        |        |        |
1        2ethnet  3serial  4        5        6        7        8        9zero    10
>
        SERIES 90-30 MODULE IN RACK 0 SLOT 6
                               SOFTWARE  CONFIGURATION
 SLOT    Catalog #:
   6

              CATALOG #      DESCRIPTION                         TYPE
           1  IC693CMM321    ETHERNET CONTROLLER                 ETHNET

         << CURSOR TO THE DESIRED CATALOG NUMBER AND PRESS THE ENTER KEY >>
         <<    PRESS PGDN KEY FOR NEXT PAGE, PGUP KEY FOR PREVIOUS PAGE   >>
C:\LM90\LESSON3                               OFFLINE
REPLACE                                      PRG: LESSON3                CONFIG VALID
```

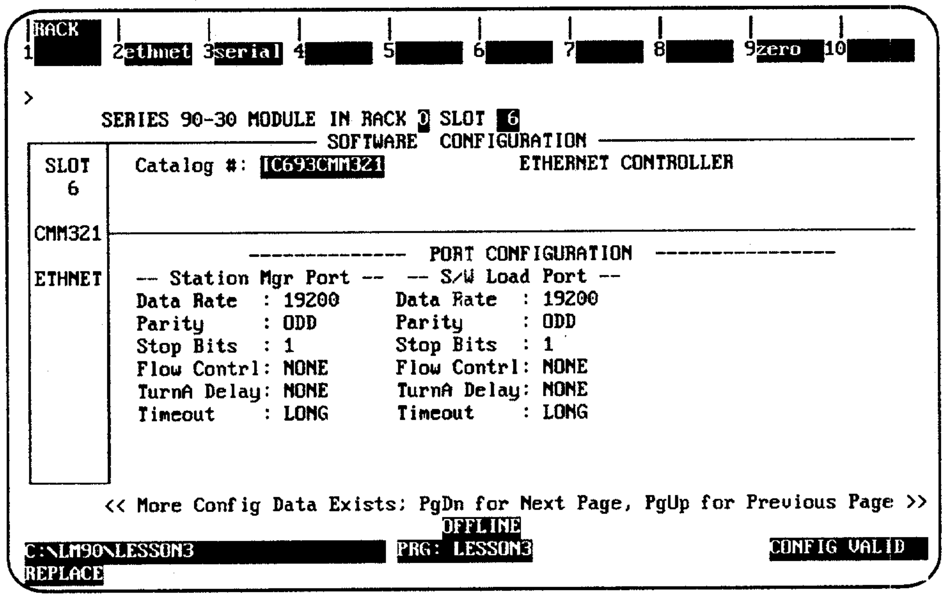
4. Press **Enter** to select the Ethernet Controller. You will then see the screen shown on the following page.



```
 RACK    |       |       |       |       |       |       |       |       |       |
1       2ethmet 3serial 4       5       6       7       8       9zero   10
>
      SERIES 90-30 MODULE IN RACK 0 SLOT 6
                           SOFTWARE  CONFIGURATION
 SLOT   Catalog #: IC693CMM321              ETHERNET CONTROLLER
  6

CMM321
        Config Mode                    :  TCP/IP
ETHNET  Status Address                 :   %I0001
        Status Length                  :   80

        IP  Address                    :  0.0.0.0
        Subnet Mask                    :  0.0.0.0
        Gateway IP Address             :  0.0.0.0
        Name Server IP Address         :  0.0.0.0

         Caution: Default IP Address (0.0.0.0) needs remote network config

      << More Config Data Exists; PgDn for Next Page, PgUp for Previous Page >>
                                      OFFLINE
C:\LM90\LESSON3                    PRG: LESSON3                    CONFIG VALID
REPLACE
```

For detailed information about this screen, refer to the first two pages of the "Procedure 2: Configuring the Ethernet Interface with Logicmaster 90-30 Configuration Software" section of chapter 2 of the *TCP/IP Ethernet Communications for the Series 90™-30 PLC* manual (GFK-1084).

5. After you have assigned the IP address, etc., press **Page Down** to display the following screen.



```
RACK
1      2ethnet 3serial 4      5      6      7      8      9zero  10

>
      SERIES 90-30 MODULE IN RACK 0 SLOT 6
                          SOFTWARE CONFIGURATION
SLOT    Catalog #: IC693CMM321           ETHERNET CONTROLLER
 6

CMM321
               -------------- PORT CONFIGURATION ---------------
ETHNET  -- Station Mgr Port --  -- S/W Load Port --
        Data Rate  : 19200      Data Rate  : 19200
        Parity     : ODD        Parity     : ODD
        Stop Bits  : 1          Stop Bits  : 1
        Flow Contrl: NONE       Flow Contrl: NONE
        TurnA Delay: NONE       TurnA Delay: NONE
        Timeout    : LONG       Timeout    : LONG

      << More Config Data Exists; PgDn for Next Page, PgUp for Previous Page >>
                                    OFFLINE
C:\LM90\LESSON3                PRG: LESSON3                     CONFIG VALID
REPLACE
```

Parameter	Description
Data Rate	Data rate (bits per second or bps) for the port. Choices are **300**, **600**, **1200**, **2400**, **4800**, **9600**, or **19200***.
Parity	Type of parity to be used on the port. Choices are **NONE**, **EVEN**, or **ODD***.
Stop Bits	Enter the number of stop bits. Choices are **1*** or **2**.
Flow Control	Type of flow control to be used on the port. Choices are **HARDWARE** or **NONE***.
Turnaround Delay	Turnaround delay time to be used for CCM on the port. Choices are **NONE***, **10 ms**, **100 ms**, or **500 ms**.
Timeout	Length of timeouts used for CCM on the port. Choices are **LONG***, **MEDIUM**, **SHORT**, or **NONE**.

* Default selection.

6. Press the **Escape** key to return to the rack display. Press **Escape** again to save the configuration.

Section 11: *Configuring a CMM Module*

Logicmaster 90-30/20/Micro programming software provides configuration support of the CMM Communications Module (multiple mode) with 32 KB. The following modes of configuration are available:

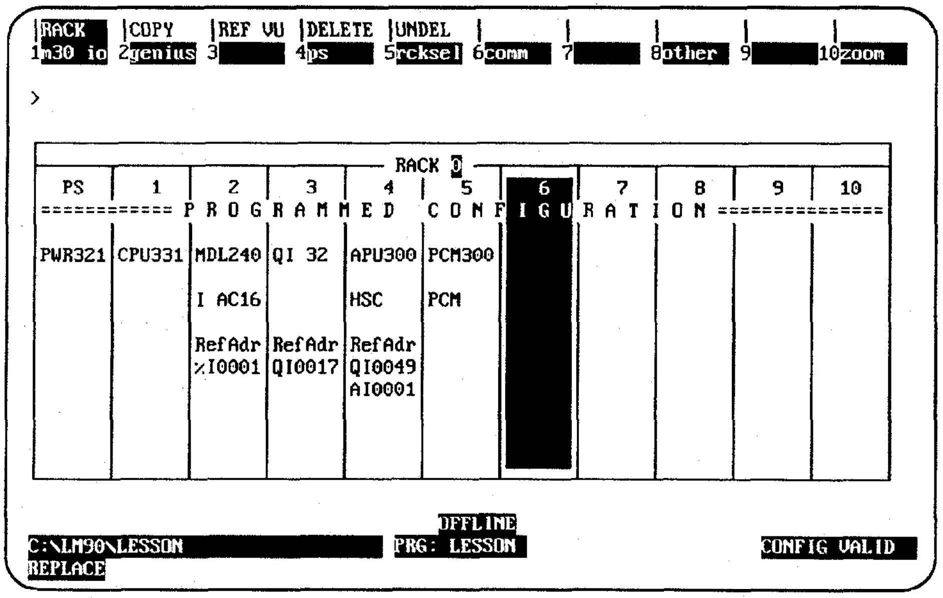
Mode	Description
CCM ONLY	Select CCM on both ports 1 and 2.
CCM/RTU	Select port 1 for CCM and port 2 for RTU.
RTU/CCM	Select port 1 for RTU and port 2 for CCM.
RTU ONLY	Select RTU on both ports 1 and 2.
SNP ONLY	Select SNP on both ports 1 and 2.
SNP/CCM	Select port 1 for SNP and port 2 for CCM.
CCM/SNP	Select port 1 for CCM and port 2 for SNP.
SNP/RTU	Select port 1 for SNP and port 2 for RTU.
RTU/SNP	Select port 1 for RTU and port 2 for SNP.

For more information on the CMM, refer to the *Series 90 Programmable Coprocessor Module and Support Software User's Manual* (GFK-0255) and the *Series 90 PLC Serial Communications User's Manual* (GFK-0582).

Configuring a CMM

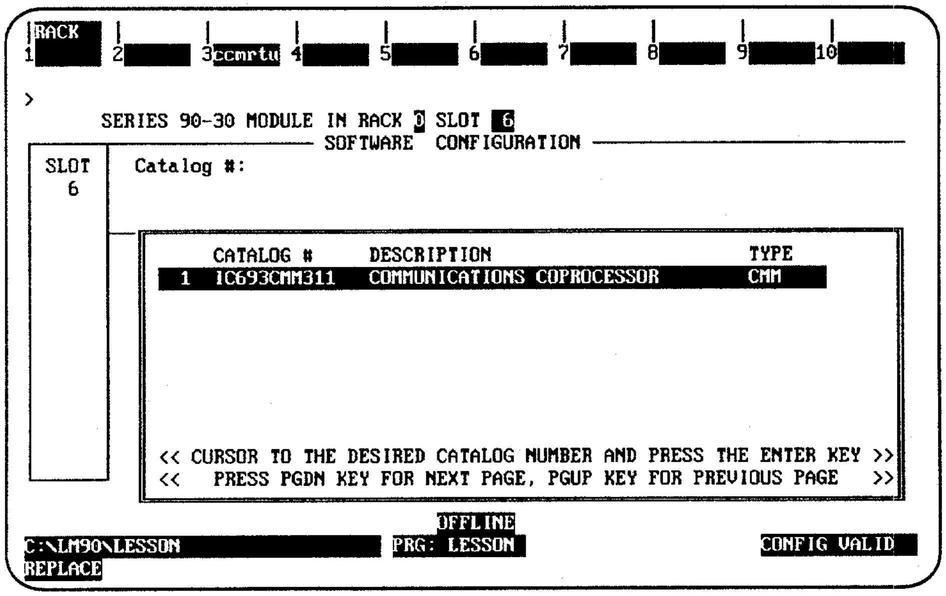
To configure a CMM on the I/O Configuration Rack screen:

1. Move the cursor to the desired rack and slot location. The slot may be either unconfigured or previously configured.



```
|RACK    |COPY    |REF VU  |DELETE  |UNDEL   |        |        |        |        |
1m30 io 2genius  3        4ps      5rcksel  6comm    7        8other   9        10zoom

>

                                     RACK 0
  PS   |  1   |  2   |  3   |  4   |  5   |  6   |  7   |  8   |  9   |  10
=========== P R O G R A M M E D   C O N F I G U R A T I O N ==============

PWR321 |CPU331|MDL240|QI 32 |APU300|PCM300|
       |      |I AC16|      |HSC   |PCM   |
       |      |RefAdr|RefAdr|RefAdr|      |
       |      |%I0001|QI0017|QI0049|      |
       |      |      |      |AI0001|      |

                                              OFFLINE
C:\LM90\LESSON                           PRG: LESSON                  CONFIG VALID
REPLACE
```

2. Press **Communications** **(F6)** and then **Serial** **(F3)** from the I/O Configuration Rack screen to display the catalog number of the CCM/RTU module.



```
RACK   |      |       |      |      |      |      |      |      |
1      2      3ccmrtu 4      5      6      7      8      9      10
>
        SERIES 90-30 MODULE IN RACK 0 SLOT 6
 ----------------------- SOFTWARE  CONFIGURATION -------------------
 SLOT | Catalog #:
  6   |
      |
      |   CATALOG #      DESCRIPTION                     TYPE
      | 1  IC693CMM311   COMMUNICATIONS COPROCESSOR       CMM
      |
      | << CURSOR TO THE DESIRED CATALOG NUMBER AND PRESS THE ENTER KEY >>
      | <<   PRESS PGDN KEY FOR NEXT PAGE, PGUP KEY FOR PREVIOUS PAGE   >>
                                   OFFLINE
C:\LM90\LESSON                   PRG: LESSON                CONFIG VALID
REPLACE
```

3. Position the cursor on the catalog number for the module you wish to select, and press the **Enter** key. The module detail screen for the module you selected will be displayed.

Selecting the Configuration Mode

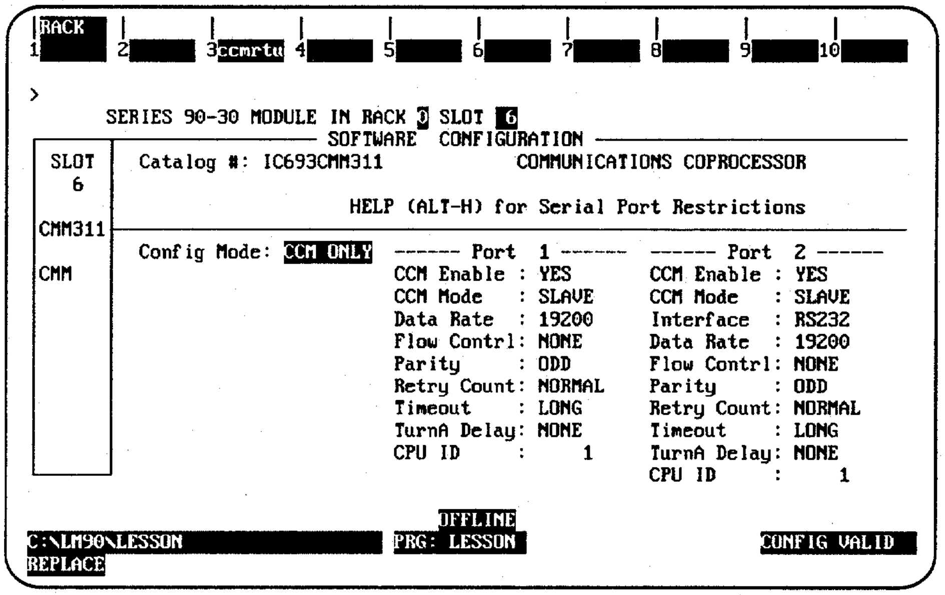
The configuration mode is selected in the *Configuration Mode* field on the module detail screen. (The module detail screen is displayed by positioning the cursor on the catalog number for the module you wish to select and pressing the **Enter** key.)

1. To select a different configuration mode, move the cursor to the *Configuration Mode* field and repeatedly press the **Tab** key until the desired mode is displayed on the screen. Then, press the **Enter** key.

2. Complete the detail screen for the desired configuration mode, and press the **Enter** key. For help selecting parameters, press **ALT-H**.

CCM ONLY Mode

When **CCM ONLY** mode is selected, the following detail screen is displayed.



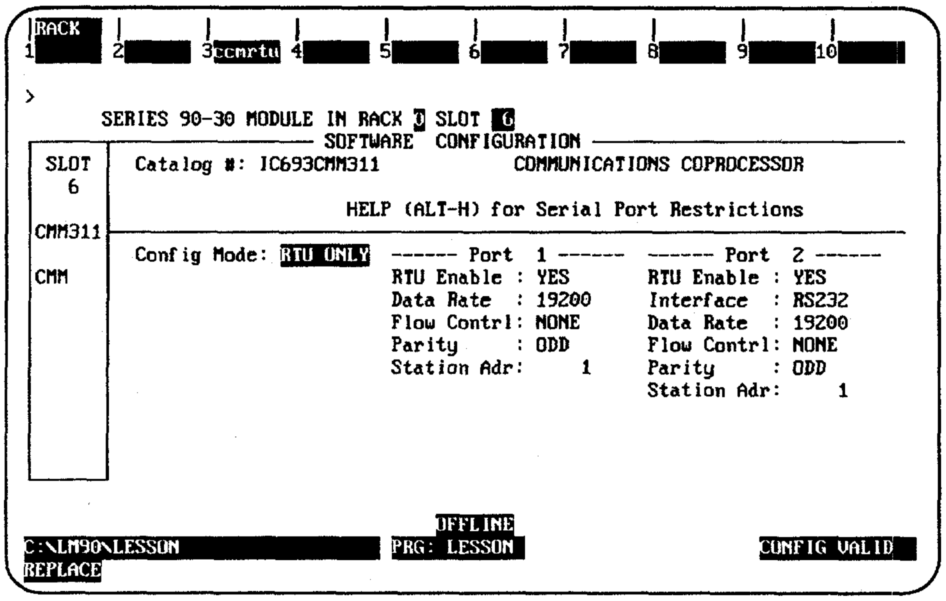
```
|RACK    |       |        |       |       |       |       |       |       |
1        2       3ccmrtu  4       5       6       7       8       9       10

>
      SERIES 90-30 MODULE IN RACK 0 SLOT 6
 ------------------------------ SOFTWARE  CONFIGURATION ------------------------
| SLOT |  Catalog #: IC693CMM311             COMMUNICATIONS COPROCESSOR
|  6   |
|      |                   HELP (ALT-H) for Serial Port Restrictions
|CMM311|-------------------------------------------------------------------------
|      |  Config Mode: CCM ONLY   ------ Port  1 ------   ------ Port  2 ------
|CMM   |                          CCM Enable : YES        CCM Enable : YES
|      |                          CCM Mode   : SLAVE      CCM Mode   : SLAVE
|      |                          Data Rate  : 19200      Interface  : RS232
|      |                          Flow Contrl: NONE       Data Rate  : 19200
|      |                          Parity     : ODD        Flow Contrl: NONE
|      |                          Retry Count: NORMAL     Parity     : ODD
|      |                          Timeout    : LONG       Retry Count: NORMAL
|      |                          TurnA Delay: NONE       Timeout    : LONG
|      |                          CPU ID     :      1     TurnA Delay: NONE
                                                          CPU ID     :      1

                                         OFFLINE
C:\LM90\LESSON                           PRG: LESSON                 CONFIG VALID
REPLACE
```

Parameter	Description
Configuration Mode	The configuration mode is set to **CCM ONLY**.
Battery Required	Specify whether a battery is required. Choices are **YES*** or **NO**.
CCM Enable	Specify whether the port is to be configured for use as a CCM port. Choices are **YES*** or **NO**.
CCM Mode	This parameter displays the availability of ports for CCM access. Choices are **SLAVE***, **PEER**, or **MASTER**.
Interface	The interface parameter for port 1 can only be **RS-232**; port 2 of the 192K and 640K modules may have an interface value of either **RS-232*** or **RS-485**. The interface parameter is not used or displayed for the 160K module.
Data Rate	Data rate (bits per second or bps) for the port. Choices are **300**, **600**, **1200**, **2400**, **4800**, **9600**, or **19200***.
Flow Control	Type of flow control to be used on the port. Choices are **HARDWARE** or **NONE***.
Parity	Type of parity to be used on the port. Choices are **NONE**, **EVEN**, or **ODD***.
Retry Count	Retry counts for CCM mode. Choices are **NORMAL*** or **SHORT**
Timeout	Length of timeouts used for CCM on the port. Choices are **LONG***, **MEDIUM**, **SHORT**, or **NONE**.
Turnaround Delay	Turnaround delay time to be used for CCM on the port. Choices are **NONE***, **10 ms**, **100 ms**, or **500 ms**.
CPU ID	Address of the port on a multi-drop network. This value is used to calculate the backoff delay upon an inquiry collision in **PEER** mode. The range of values allowed in this field is **1*** to **254**.

* Default selection.

RTU ONLY Mode

When **RTU ONLY** mode is selected, the following detail screen is displayed.



```
RACK
1     2     3ccmrtu 4     5     6     7     8     9     10
>
      SERIES 90-30 MODULE IN RACK 3 SLOT 6
              ----------- SOFTWARE  CONFIGURATION -----------
SLOT  | Catalog #: IC693CMM311          COMMUNICATIONS COPROCESSOR
 6    |
      |            HELP (ALT-H) for Serial Port Restrictions
CMM311|------------------------------------------------------
      | Config Mode: RTU ONLY  ------ Port  1 ------  ------ Port  2 ------
CMM   |                        RTU Enable : YES       RTU Enable : YES
      |                        Data Rate  : 19200     Interface  : RS232
      |                        Flow Contrl: NONE      Data Rate  : 19200
      |                        Parity     : ODD       Flow Contrl: NONE
      |                        Station Adr:     1     Parity     : ODD
      |                                               Station Adr:     1

                                      OFFLINE
C:\LM90\LESSON                        PRG: LESSON               CONFIG VALID
REPLACE
```

Parameter	Description
Configuration Mode	The configuration mode is set to **CCM ONLY**.
RTU Enable	Specify whether the port is to be configured for use as an RTU port. Choices are **YES*** or **NO**.
Interface	The interface parameter for port 1 can only be **RS-232**; port 2 of the 192K and 640K modules may have an interface value of either **RS-232*** or **RS-485**. The interface parameter is not used or displayed for the 160K module.
Data Rate	Data rate (bits per second or bps) for the port. Choices are **300**, **600**, **1200**, **2400**, **4800**, **9600**, or **19200***.
Flow Control	Type of flow control to be used on the port. Choices are **HARDWARE** or **NONE***.
Parity	Type of parity to be used on the port. Choices are **NONE**, **ODD***, or **EVEN**.
Station Address	Enter a value from **1*** to **247**.

* Default selection.

CCM/RTU Mode

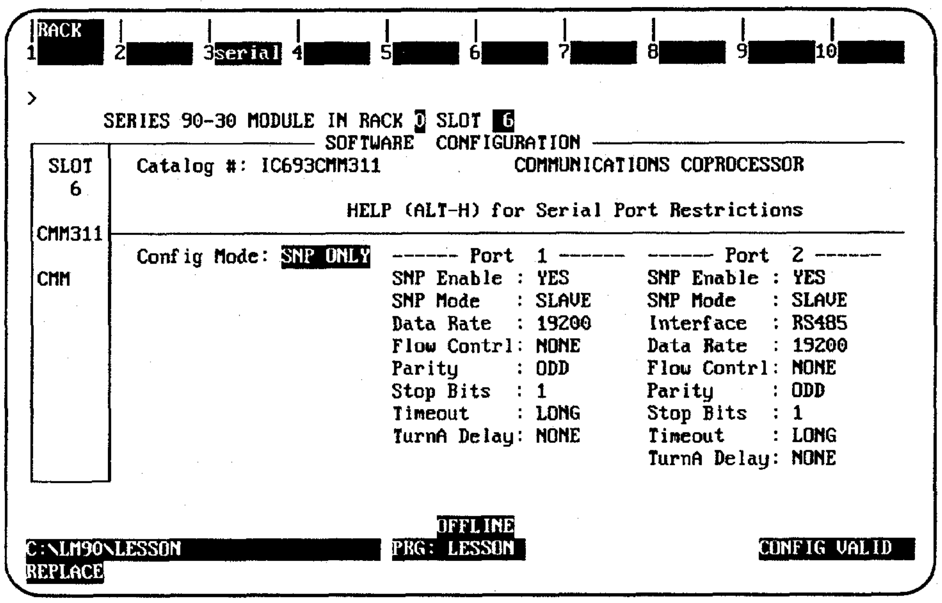
CCM/RTU mode selects port 1 for CCM and port for RTU. The parameters for port 1 are the same as for **CCM ONLY** mode; and for port 2, the parameters are the same as for **RTU ONLY** mode.

RTU/CCM Mode

RTU/CCM mode selects port 1 for RTU and port 2 for CCM. The parameters for port 1 are the same as for **RTU ONLY** mode; and for port 2, the parameters are the same as for **CCM ONLY** mode.

SNP ONLY Mode

When **SNP ONLY** mode is selected, the following detail screen is displayed.



```
 RACK
 1      2      3serial 4      5      6      7      8      9      10
>
     SERIES 90-30 MODULE IN RACK 0 SLOT 6
                        SOFTWARE  CONFIGURATION
 SLOT   Catalog #: IC693CMM311            COMMUNICATIONS COPROCESSOR
  6
                          HELP (ALT-H) for Serial Port Restrictions
CMM311
        Config Mode: SNP ONLY  ------ Port  1 ------    ------ Port  2 ------
CMM                           SNP Enable : YES         SNP Enable : YES
                              SNP Mode   : SLAVE       SNP Mode   : SLAVE
                              Data Rate  : 19200       Interface  : RS485
                              Flow Contrl: NONE        Data Rate  : 19200
                              Parity     : ODD         Flow Contrl: NONE
                              Stop Bits  : 1           Parity     : ODD
                              Timeout    : LONG        Stop Bits  : 1
                              TurnA Delay: NONE        Timeout    : LONG
                                                       TurnA Delay: NONE

                                          OFFLINE
C:\LM90\LESSON                            PRG: LESSON              CONFIG VALID
REPLACE
```

Parameter	Description
Configuration Mode	The configuration mode is set to **SNP ONLY**.
SNP Enable	Specify whether the port is to be configured for use as an SNP port. Choices are **YES*** or **NO**.
SNP Mode	This parameter displays the availability of ports for SNP access. Choices are **SLAVE*** or **MASTER**.
Interface	The interface parameter can be **RS-232*** or **RS-485**.
Data Rate	Data rate (bits per second or bps) for the port. Choices are **300**, **600**, **1200**, **2400**, **4800**, **9600**, or **19200***.
Parity	Type of parity to be used on the port. Choices are **NONE**, **ODD***, or **EVEN**.
Stop Bits	Most serial communication uses at least one stop bit. Slower devices use two stop bits. Choices are **1*** or **2**.
Flow Control	Type of flow control to be used on the port. Choices are **NONE*** or **HARDWARE**.
Turnaround Delay	Turnaround delay time to be used for CCM on the port. Choices are **NONE***, **10 ms**, **100 ms**, or **500 ms**.
Timeout	Length of timeouts used for SNP on the port. Choices are **LONG***, **MEDIUM**, **SHORT**, or **NONE**.

* Default selection.

SNP/CCM Mode

SNP/CCM mode selects port 1 for SNP and port 2 for CCM. The parameters for port 1 are the same as for **SNP ONLY** mode; and for port 2, the parameters are the same as for **CCM ONLY** mode.

CCM/SNP Mode

CCM/SNP mode selects port 1 for CCM and port 2 for SNP. The parameters for port 1 are the same as for **CCM ONLY** mode; and for port 2, the parameters are the same as for **SNP ONLY** mode.

SNP/RTU Mode

SNP/RTU mode selects port 1 for SNP and port 2 for RTU. The parameters for port 1 are the same as for **SNP ONLY** mode; and for port 2, the parameters are the same as for **RTU ONLY** mode.

RTU/SNP Mode

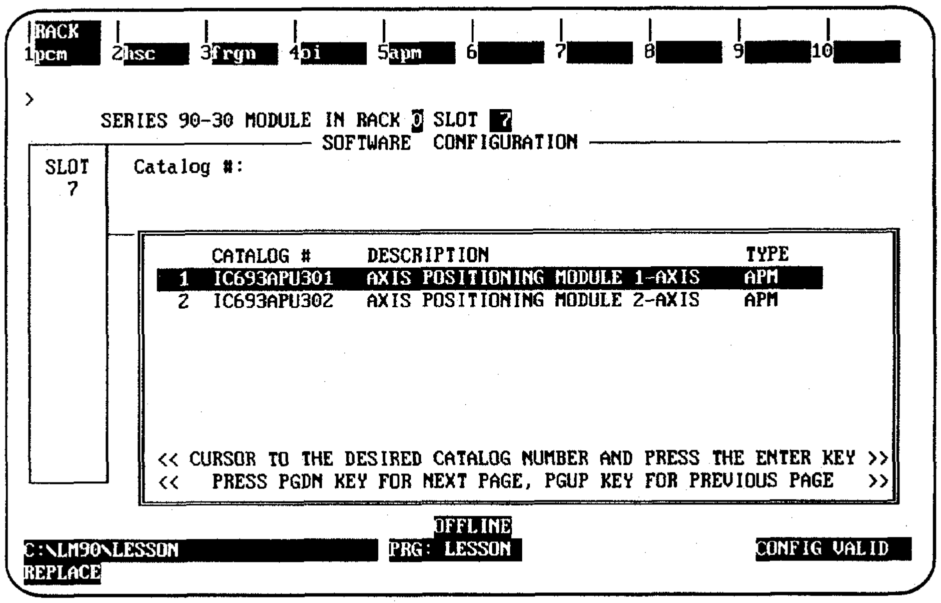
RTU/SNP mode selects port 1 for RTU and port 2 for SNP. The parameters for port 1 are the same as for **RTU ONLY** mode; and for port 2, the parameters are the same as for **SNP ONLY** mode.

Section 12: Configuring an APM

The Axis Positioning Module (APM) is an intelligent, fully-programmable, one or two-axis positioning controller integrated into the Series 90-30 PLC system. You must have a Release 3 or later CPU module in order to configure a slot in the I/O rack for an APM. For more information on the APM, refer to the *Series 90-30 Axis Positioning Module Quick Reference and Installation Guide*, GFK-0707, and the *Series 90-30 Axis Positioning Module Programmer's Manual*, GFK-0664.

To configure an Axis Positioning Module on the I/O Configuration Rack screen:

1. Move the cursor to the desired rack and slot location. The slot may be either configured or previously unconfigured.
2. Press **Other** (**F8**) and then **apm** (**F5**) from the I/O Configuration Rack screen to display the catalog number of the APM.



```
 RACK
1pcm    2hsc    3frgn   4pi     5apm    6       7       8       9       10
>
     SERIES 90-30 MODULE IN RACK 0 SLOT 7
                          SOFTWARE  CONFIGURATION
 SLOT   Catalog #:
  7
                 CATALOG #      DESCRIPTION                      TYPE
              1  IC693APU301    AXIS POSITIONING MODULE 1-AXIS   APM
              2  IC693APU302    AXIS POSITIONING MODULE 2-AXIS   APM

          << CURSOR TO THE DESIRED CATALOG NUMBER AND PRESS THE ENTER KEY >>
          <<   PRESS PGDN KEY FOR NEXT PAGE, PGUP KEY FOR PREVIOUS PAGE   >>
                                        OFFLINE
C:\LM90\LESSON                         PRG: LESSON                CONFIG VALID
REPLACE
```

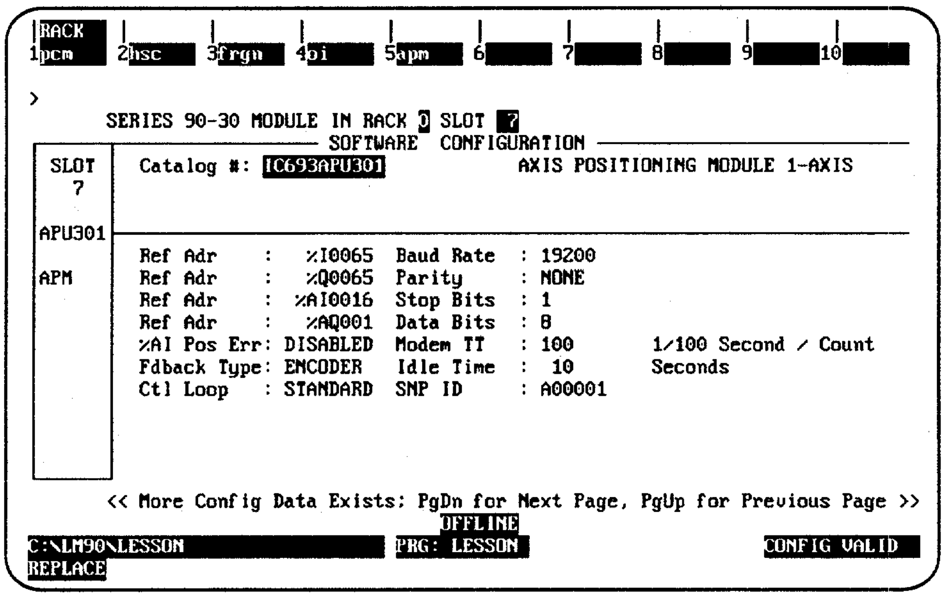
Note

The instructions on the following pages describe how to configure a slot in the I/O rack for a single axis APM. These instructions also apply to the two-axis APM as well. The only difference is in the catalog number you select (IC693APU302 for the two-axis APM instead of IC693APU301 for the single axis APM).

Configuring a Single Axis APM

To configure a single axis APM:

1. Position the cursor on the catalog number for the single axis APM (IC693APU301), and press the **Enter** key. The following detail screen is displayed.



```
|RACK |     |     |     |     |     |     |     |     |
1pcm  2hsc  3frgn 4oi   5apm  6     7     8     9     10

>
      SERIES 90-30 MODULE IN RACK 0 SLOT 7
                     SOFTWARE  CONFIGURATION
 SLOT   Catalog #: IC693APU301        AXIS POSITIONING MODULE 1-AXIS
  7
APU301
        Ref Adr     :   %I0065  Baud Rate  : 19200
APM     Ref Adr     :   %Q0065  Parity     : NONE
        Ref Adr     :  %AI0016  Stop Bits  : 1
        Ref Adr     :   %AQ001  Data Bits  : 8
        %AI Pos Err: DISABLED  Modem TT   : 100       1/100 Second / Count
        Fdback Type: ENCODER   Idle Time  :  10       Seconds
        Ctl Loop    : STANDARD  SNP ID     : A00001

      << More Config Data Exists: PgDn for Next Page, PgUp for Previous Page >>
                                          OFFLINE
C:\LM90\LESSON                            PRG: LESSON                CONFIG VALID
REPLACE
```

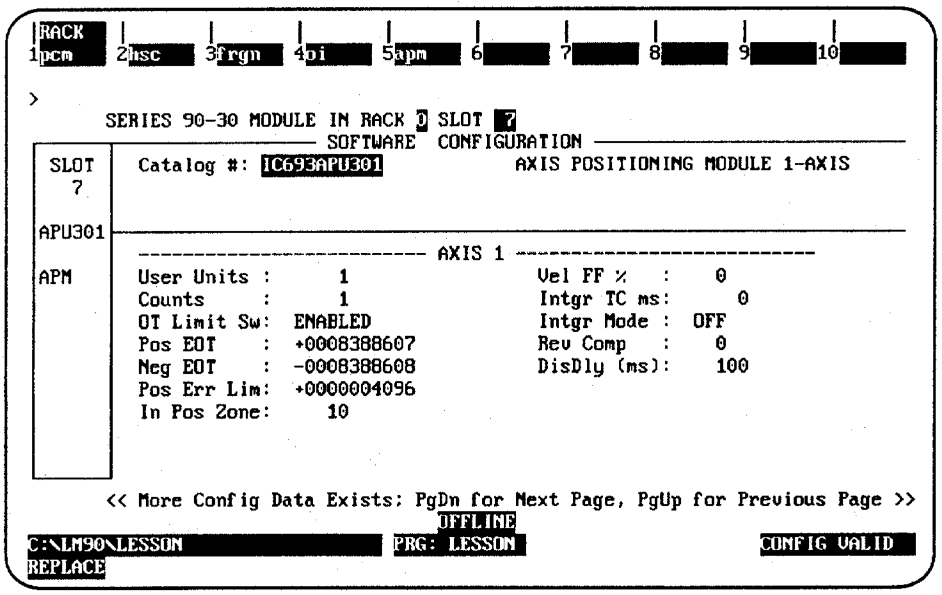
Parameter	Description
Reference Address	The starting address for the %I user reference. Default= **%I0001** or the next highest available address. For both the single axis APM and the two-axis APM, the reference length is fixed at **32** bits.
Reference Address	The starting address for the %Q user reference. Default= **%Q0001** or the next highest available address. For both the single axis APM and the two-axis APM, the reference length is fixed at **32** bits.
Reference Address	The starting address for the %AI user reference. Default = **%AI001** or the next highest available address. For the single axis APM, the reference length is fixed at **15** words. For the two-axis APM, the reference length is fixed at **28** words.
Reference Address	The starting address for the %AQ user reference. Default = **%AQ001** or the next highest available address. For both the single axis APM, and the two-axis APM, the reference length is fixed at **6** words.
%AI Position Error	If **DISABLED***, the command position is displayed in the %AI table. If **ENABLED**, the position error is displayed in the %AI table.
Feedback Type	**ENCODER*** selects A quad B (x4) incremental encoder input mode. **LINEAR** selects Temposonics linear transducer (absolute feedback) input mode. **RESOLVER** selects single or multiple resolver (absolute feedback) input mode. **CUSTOM1** and **CUSTOM2** configure the APM inputs for special applications.
Control Loop	**STANDARD*** selects the normal APM motion control loop. The **STANDARD** loop provides a velocity command output proportional to position error, with optional velocity feed forward and integrator gain terms. **CUSTOM** loops are individually designed for special applications. For the two-axis APM, **FOLLOWER** selects a control loop which allows ratio tracking of a master input with zero following error.

* Default selection.

Parameter	Description
Baud Rate	Transmission rate (in bits per second or bps) of data through the SNP port. Choices are **300, 600, 1200, 2400, 4800, 9600,** or **19200***.
Parity	Parity is indicated by an **ODD*** or **EVEN** number of bits, or whether no parity bit (**NONE**) is added to the word.
Stop Bits	Number of stop bits. Most serial communications use one stop bit. Slower devices may use two stop bits. Choices are **1*** or **2**.
Data Bits	Number of data bits. Specify whether the CPU recognizes **7** or **8*** bit words.
Modem Turnaround Time	The time required for the modem to start data transmission after receiving the transmit request. Values are **0*** to **2550**, in multiples of **10** milliseconds.
Idle Time	Maximum link idle time. The time the module should wait for the next message to be received from the communicating device before it believes that the program device has failed and proceeds to its base state. Values are **1** to **60** seconds. (Default = **10** seconds)
SNP ID	The identification number of the SNP port. Messages for the module are sent to this address. (Default = **A00001**)

* Default selection.

2. Press the **Page Down** key to display the following detail screen:



```
|RACK  |      |      |      |      |      |      |      |      |
1pcm   2hsc   3frgn  4bi    5apm   6      7      8      9      10

>
      SERIES 90-30 MODULE IN RACK 0 SLOT 7
               ———— SOFTWARE  CONFIGURATION ————————————————
 SLOT   Catalog #: IC693APU301          AXIS POSITIONING MODULE 1-AXIS
  7

APU301
        ------------------------ AXIS 1 ------------------------
APM     User Units :      1            Vel FF %    :    0
        Counts     :      1            Intgr TC ms:       0
        OT Limit Sw:  ENABLED          Intgr Mode :   OFF
        Pos EOT    :  +0008388607      Rev Comp    :    0
        Neg EOT    :  -0008388608      DisDly (ms):    100
        Pos Err Lim:  +0000004096
        In Pos Zone:     10

      << More Config Data Exists: PgDn for Next Page, PgUp for Previous Page >>
                                     OFFLINE
C:\LM90\LESSON                     PRG: LESSON                 CONFIG VALID
REPLACE
```

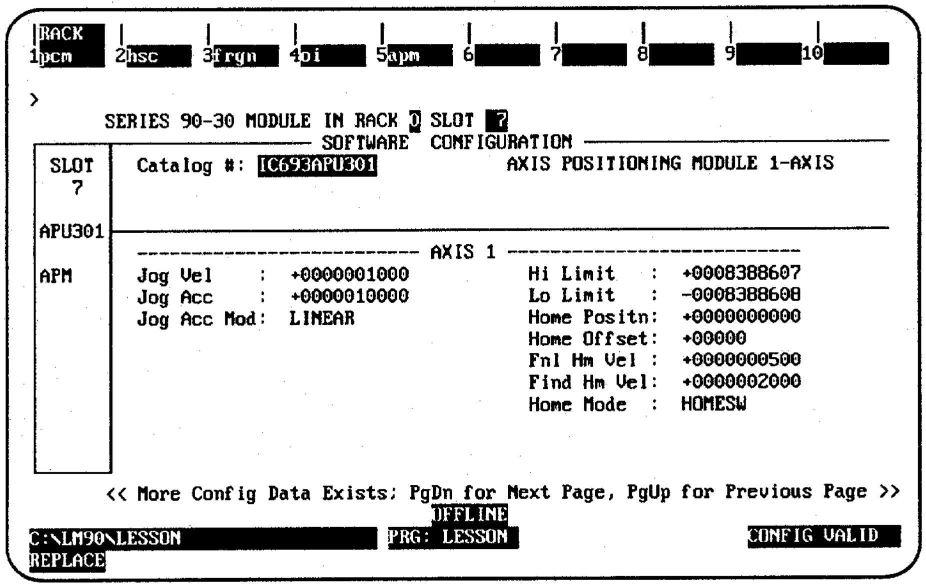
Parameter	Description
User Units	Scale factor which allows the APM to be programmed in units appropriate for the application. The ratio of user units to counts must be in the range **32:1** to **1:32**. Values are **1*** to **+65,535**.
Counts	Scale factor which allows the APM to be programmed in units appropriate for the application. The ratio of user units to counts must be in the range **32:1** to **1:32**. Values are **1*** to **+65,535**.

* Default selection.

Parameter	Description
Overtravel Limit Switch	Specify whether the APM uses the hardware overtravel limit switch inputs (CTL05 and CTL06 for axis 1; CTL07 and CTL08 for axis 2). If **ENABLED**, then 10 - 30 VDC must be applied to the inputs in order for the APM to operate. Values are **ENABLED*** or **DISABLED**.
Positive Software End of Travel Limit	If the APM is commanded to go to a position greater than the positive EOT, an error will result and the APM will not allow axis motion. Values are **-8,388,608** to **+8,388,607*** user units.
Negative Software End of Travel Limit	If the APM is commanded to go to a position less than the negative EOT, an error will result and the APM will not allow axis motion. Values are **0** to **-8,388,608*** user units.
Position Error Limit	The maximum position error (Commanded Position – Actual Position) allowed when the APM is controlling a servo. This parameter should normally be set to a value 10 to 20 percent higher than the highest position error encountered under normal servo operations. Range = **256*** (user units / counts) <= POS ERR LIM <= **60,000*** (user units / counts). (Default = **4096**)
In Position Zone	When the servo position error is within this value and no motion is commanded, the **IN ZONE** status bit is set. This parameter also determines the position error at which **PMOVE**s are considered to be complete. Range = **0** to **2000**. (Default = **10**)
Position Loop Time Constant	The desired servo position loop time constant (in milliseconds). The lower the value, the faster the system's response. Values which are too low will cause system instability and oscillation. For accurate tracking of the commanded velocity profile, the position loop time constant should be 1/4 to 1/2 of the minimum system deceleration time. The time will not be accurate unless the velocity at 10V value is set correctly. Values are **20** to **+32,767** ms. (Default = **1000**)
Velocity at 10V	Actual servo velocity (in user units per second) for an APM velocity command output of 10V. This value must be configured correctly in order for the position loop time constant and the velocity feedforward gain percent factor to be accurate. The APMs **FORCE D/A OUTPUT** %AQ immediate command and the **ACTUAL VELOCITY** %AI return data can be used to determine the proper configuration value. Values are **100** to **+8,388,607**. (Default = **4000**)
Velocity Feedforward Gain	The percentage of commanded velocity that is added to the APM velocity command output. Increasing this parameter causes the servo to operate with faster response and reduced position error. Values are **0*** to **100** percent; optimum values are **80** to **90** percent. The velocity at 10V value must be set correctly for proper operation of the velocity feedforward gain factor.
Integrator Time Constant (in milliseconds)	Values should be 5 to 10 times the position loop time constant setting. Servo instability will result if the integrator time constant is set too low. When the integrator mode is set to **CONTINUOUS**, deceleration times must be 5 to 10 times longer than the integrator time constant setting or servo overshoot will occur. Setting the time constant to zero turns off the integrator. (Default = **0**)
Integrator Mode	Operating mode for the position error integrator. **OFF*** means that the integrator is not used. **CONTINU** means the integrator runs continuously, even during servo motion. **IN ZONE** means the integrator only runs when the servo is at rest.
Reversal Compensation	A compensation factor which allows the servo to reverse direction and still provide accurate positioning in systems containing backlash. Values are **0*** to **+255**.
Servo Drive Disable Delay	The time delay (in milliseconds) from zero velocity command to the drive enable output switching off. Disable delay is effective when the **ENABLE DRIVE %I** bit is turned off or certain error conditions occur. This time delay should be longer than the deceleration time of the servo from maximum speed. Values are **0** to **+32,767** ms. (Default = **100**)

* Default selection.

3. Press the **Page Down** key again to display this detail screen:

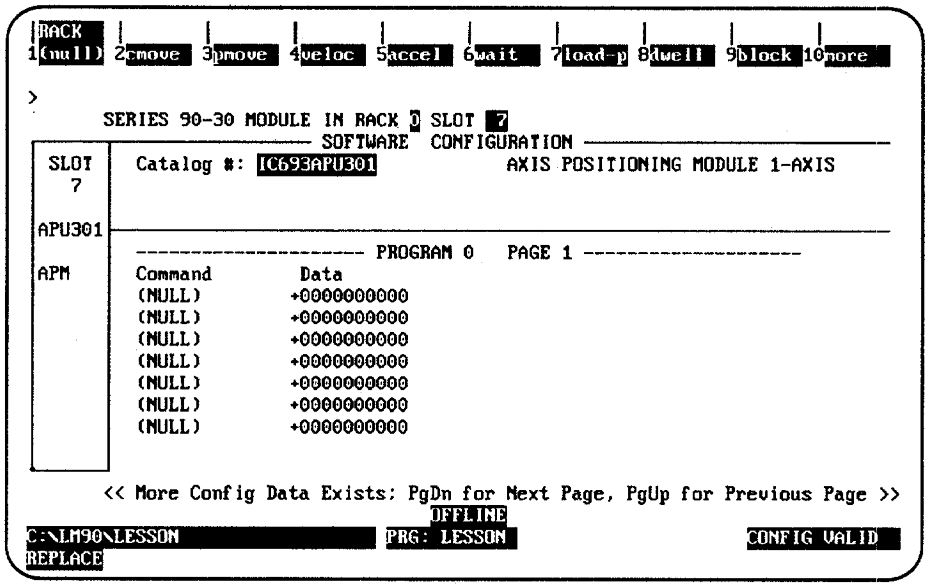


```
RACK
1pcm   2hsc   3frgn   4di   5apm   6   7   8   9   10

>
      SERIES 90-30 MODULE IN RACK 0 SLOT 7
                         SOFTWARE CONFIGURATION
 SLOT   Catalog #: IC693APU301            AXIS POSITIONING MODULE 1-AXIS
   7

APU301
                          AXIS 1
APM     Jog Vel     :  +0000001000        Hi Limit    :  +0008388607
        Jog Acc     :  +0000010000        Lo Limit    :  -0008388608
        Jog Acc Mod:  LINEAR              Home Positn:  +0000000000
                                          Home Offset:  +00000
                                          Fnl Hm Vel :  +0000000500
                                          Find Hm Vel:  +0000002000
                                          Home Mode  :  HOMESW

      << More Config Data Exists; PgDn for Next Page, PgUp for Previous Page >>
                                   OFFLINE
C:\LM90\LESSON                     PRG: LESSON                CONFIG VALID
REPLACE
```

Parameter	Description
Jog Velocity	The velocity (in user units per second) at which the servo moves during a jog operation. Values are **0** to **+8,388,607**. (Default = **1000**)
Jog Acceleration Rate	The acceleration rate (in user units per second) used during jog, find home, move at velocity, and abort operations. Values are **0** to **+8,388,607**. (Default = **10000**)
Jog Acceleration Mode	The acceleration mode for jog, find home, move at velocity, and abort operations. **LINEAR*** causes commanded velocity to change linearly with time. **SCURVE** causes commanded velocity to change more slowly than the linear mode at the beginning and end of acceleration intervals.
High Count Limit	When the axis is moving in the positive direction and this value is reached, actual position will roll over to the low count limit. (Default = **8,388,607**)
Low Count Limit	When the axis is moving in the negative direction and this value is reached, actual position will roll over to the high count limit. (Default = **-8,388,607**)
Home Position	The value (in user units) assigned to actual position at the end of a find home cycle. Values are **-8,388,608** to **+8,388,607**. (Default = **0**)
Home Position Offset	The offset (in user units) of the servo final stopping point at the completion of a find home cycle. Home offset adjusts the final servo stopping point relative to the encoder marker. Values are **-32,768** to **+32,767**. (Default = **0**)
Final Home Velocity	The velocity (in user units per second) at which the servo seeks the final home switch transition and encoder marker pulse at the end of a find home cycle. Final home velocity must be slow enough to allow a 5 millisecond delay between the final home switch transition and the encoder marker pulse. Values are **0** to **+8,388,607**. (Default = **500**)
Find Home Velocity	The velocity (in user units per second) at which the servo seeks the initial home switch transitions during the find home cycle. If desired, find home velocity can be set to a high value to allow the servo to quickly locate the home switch. Values are **0** to **+8,388,607**. (Default = **2000**)
Home Mode	Select **HOMESW** if a home switch input is used as part of the find home cycle. Select **MOVE+** or **MOVE-** for a unidirectional home cycle which does not use the home switch input.

* Default selection.

4. If you press the **Page Down** key again, the following screen is displayed.



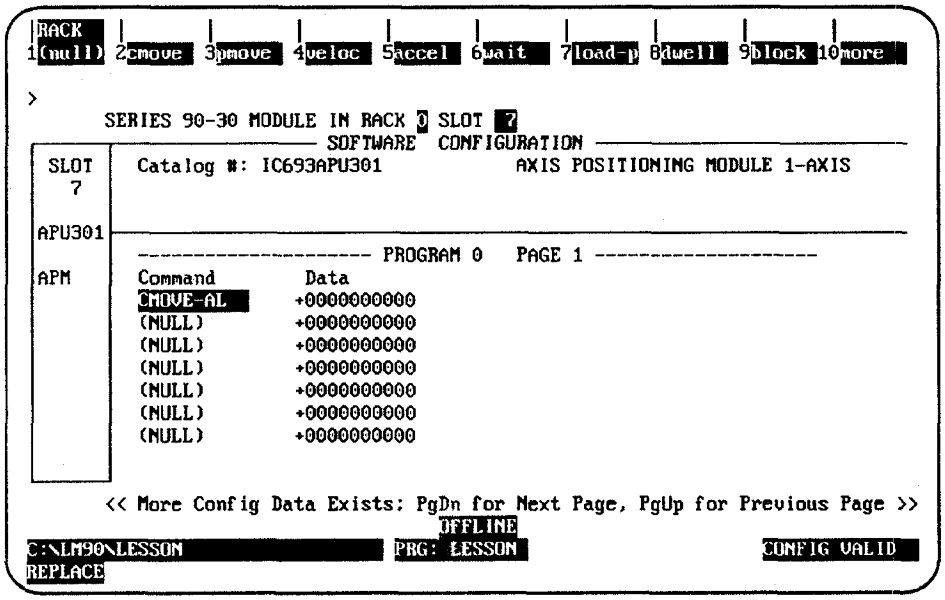
```
RACK |        |        |        |        |        |        |        |        |
1(null) 2cmove 3pmove 4veloc 5accel 6wait 7load-p 8dwell 9block 10more

>
        SERIES 90-30 MODULE IN RACK 0 SLOT 7
 ______________________ SOFTWARE  CONFIGURATION _________________________
| SLOT |  Catalog #: IC693APU301          AXIS POSITIONING MODULE 1-AXIS
|  7   |
|APU301|__________________________________________________________________
|      |  -------------------- PROGRAM 0   PAGE 1 -------------------
|APM   |  Command         Data
|      |  (NULL)          +0000000000
|      |  (NULL)          +0000000000
|      |  (NULL)          +0000000000
|      |  (NULL)          +0000000000
|      |  (NULL)          +0000000000
|      |  (NULL)          +0000000000
|      |  (NULL)          +0000000000
|______|
       << More Config Data Exists: PgDn for Next Page, PgUp for Previous Page >>
                                        OFFLINE
C:\LM90\LESSON                        PRG: LESSON                 CONFIG VALID
REPLACE
```

This screen, and subsequent screens like this, enable you to specify a small set of instructions to be downloaded to the APM upon initialization. (Note the change of function softkey assignments on these screens.) The following table defines these instructions; subsequent tables provide more detailed information about each instruction.

Command	Description
NULL	The Null command causes no action.
CMOVE	The Continuous Move (data = user units) command causes servo motion without requiring the servo to stop in order for the next command to execute.
PMOVE	The Positioning Move (data = user units) command causes the servo t come to rest and be in zone before executing the next command.
VELOC	The velocity (in user units per second) for subsequent CMOVE and PMOVE commands.
ACCEL	The acceleration rate (in user units per second) for subsequent CMOVE and PMOVE commands.
WAIT	The Wait for Discrete Input command causes the APM to wait for a bit number (1 – 12) to be set before executing the next command.
LOAD-P	The Load Parameter command loads the specified parameter data into one of the 20 parameter registers.
DWELL	The Dwell command causes a delay equal to the dwell time (in milliseconds) before execution of the next command.
BLOCK	The Block Number command assigns the specified block number to subsequent commands. The block number is reported in the APM %AI input data.
JUMP	The Jump command causes program execution to transfer to the specified block number. The Jump can be unconditional (always occurs) or conditional (occurs when the specified CTL bit is set). For a conditional Jump, the CTL bit is tested when the Jump command is first encountered. If the CTL bit is not set, testing of the bit will occur every 2 milliseconds during subsequent PMOVE, CMOVE, or Dwell commands. Testing continues until the CTL bit is set or a subsequent Block number command is encountered in the motion program.

Field	Description
Data	This field contains either a 24-bit signed integer or the number of an APM register if the command has a `-P` extension.
Command	A legal APM command. For configuration, the legal commands are listed below.
	(NULL): Do nothing. **BLOCK:** Define block number. Values are `1` to `+65,635`. (Default = `1`)
	The following commands have a value range from `-8,388,608` to `+8,388,607`. When the `-P` suffix is used on the command, however, the maximum data value is `255`. (Default = `1`) (A `-1` placed at the end of the command indicates that the command is for axis 1; a `-2` indicates axis 2.)
	CMOVE-AL: Regular move, absolute, linear. **CMOVE-AS:** Regular move, absolute, s-curve. **CMOVE-AS-P:** Regular move, absolute, s-curve, use data in APM register. **CMOVE-IL:** Regular move, incremental, linear. **CMOVE-IL-P:** Regular move, incremental, linear, use data in APM register. **CMOVE-IS:** Regular move, incremental, s-curve. **CMOVE-IS-P:** Regular move, incremental, s-curve, use data in APM register. **PMOVE-AL:** Positioning move, absolute, linear. **PMOVE-AL-P:** Positioning move, absolute, linear, use data in APM register. **PMOVE-AS:** Positioning move, absolute, s-curve. **PMOVE-AS-P:** Positioning move, absolute, s-curve, use data in APM register. **PMOVE-IL:** Positioning move, incremental, linear. **PMOVE-IL-P:** Positioning move, incremental, linear, use data in APM register. **PMOVE-IS:** Positioning move, incremental, s-curve **PMOVE-IS-P:** Positioning move, incremental s-curve, use data in APM register.
	The following commands have a value range of `1` to `98,388,607`. When the `-P` suffix is used on the command, however, the maximum data value is `255`. (Axis 1 = `-1`; Axis 2 = `-2`) **VELOC:** Set velocity. (Default = `2000`) **VELOC-P:** Set velocity to data in APM register. (Default = `1`)
	The following commands have a value range of `1` to `134,217,727`. When the `-P` suffix is used on the command, however, the maximum data value is `255`. (Axis 1 = `-1`; Axis 2 = `-2`) **ACCEL:** Set acceleration. (Default = `5000`) **ACCEL-P:** Set acceleration to data in APM register. (Default = `1`)
	WAIT: Wait for some bit to go high before moving. Values are `1`, `2`, `4`, `16`, etc. (Default = `1`)
	The following commands have a value range from `-8,388,608` to `+8,388,607`. (Default = `0`) **LOAD-P01:** Load APM parameter register number 1. **LOAD-P02:** Load APM parameter register number 2. • • • **LOAD-P20:** Load APM parameter register number 20.
	The following commands have a value range from `0` to `+65,535`. When the `-P` suffix is used on the command, however, the maximum data value is `255`. (Axis 1 = `-1`; Axis 2 = `-2`) **DWELL:** Wait X milliseconds. Default = `0`) **DWELL-P:** Wait X milliseconds, where X is the value of the APM register. (Default = `1`)
	If an entry other than one of the legal commands listed above is made, Logicmaster 90-30/90-20 software will display the error message, "Incorrect data entry."
Axis Number	The axis number may be `1`, `2`, or `N/A` (Not Applicable).

5. Using the list of commands above, enter the command in the *Command* field, or use the function softkeys to select a command. To use the softkeys, move the cursor to the appropriate command field and press the softkey for the desired command. In the following example, **CMOVE** `(F2)` was pressed.



```
 RACK
1(null) 2cmove 3pmove 4veloc 5accel 6wait 7load-p 8dwell 9block 10more
>
        SERIES 90-30 MODULE IN RACK 0 SLOT 7
                          SOFTWARE  CONFIGURATION
 SLOT   Catalog #: IC693APU301           AXIS POSITIONING MODULE 1-AXIS
  7

APU301
        --------------------- PROGRAM 0   PAGE 1 -------------------
APM     Command         Data
        CMOVE-AL        +0000000000
        (NULL)          +0000000000
        (NULL)          +0000000000
        (NULL)          +0000000000
        (NULL)          +0000000000
        (NULL)          +0000000000
        (NULL)          +0000000000

        << More Config Data Exists: PgDn for Next Page, PgUp for Previous Page >>
                                         OFFLINE
C:\LM90\LESSON                    PRG: LESSON                    CONFIG VALID
REPLACE
```

6. If there is more than one type of command for a particular command (e.g., CMOVE), use the Tab key to move through the various selections. In the example, CMOVE-AL is the default for the **CMOVE** `(F2)` softkey. When F2 is initially pressed, CMOVE-AL is displayed. To select another CMOVE command in this field, press the **Tab** key to display CMOVE-AL-P, CMOVE-AS, CMOVE-AS-P, CMOVE-IL, CMOVE-IL-P, CMOVE-IS, and CMOVE-IS-P, in order.

7. When you press **Load** `(F7)` for the LOAD command, LOAD-P01 is initially displayed. To change the APM register number, press the **Tab** key. The display will change from LOAD-P01 to LOAD-P02 ..... LOAD-Pn (where n is the parameter register number). This eliminates having to make an entry in the *Data* field for the LOAD command. If a value is entered into the data register for the LOAD command, it is simply ignored.

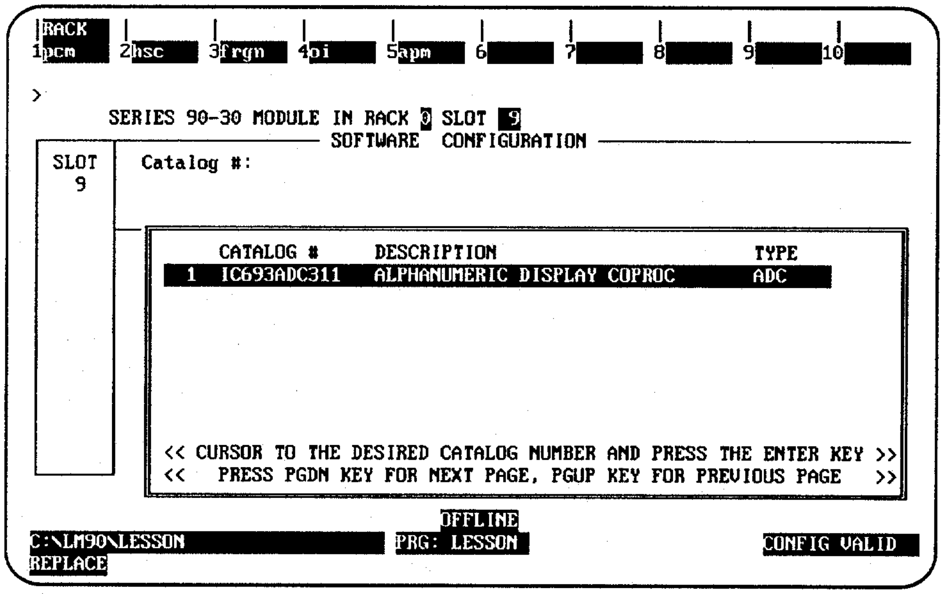
8. The other commands do not support the **Tab** key. If the **Tab** key is pressed, an "Inactive Key" error message is displayed and the key is ignored. In addition, the **Tab** key will also generate an "Inactive Key" error message if it is pressed in the *Data* field.

Section 13: Configuring an ADC Module

The Alphanumeric Display Coprocessor (ADC) Module is used with the CIMPLICITY 90-ADS software for developing user screens and graphic displays. The CIMPLICITY 90-ADS system is a character-based operator interface generator system for use with the Series 90-30 PLC. For more information on this module, refer to the *CIMPLICITY 90-ADS User's Manual*, GFK-0499.

To configure this module on the I/O Configuration Rack screen:

1. Move the cursor to the desired rack and slot location. The slot may be either unconfigured or previously configured.
2. Press **Other** (**F8**) and then **Operator Interface** (**F4**) from the I/O Configuration Rack screen to display the catalog number for the module.



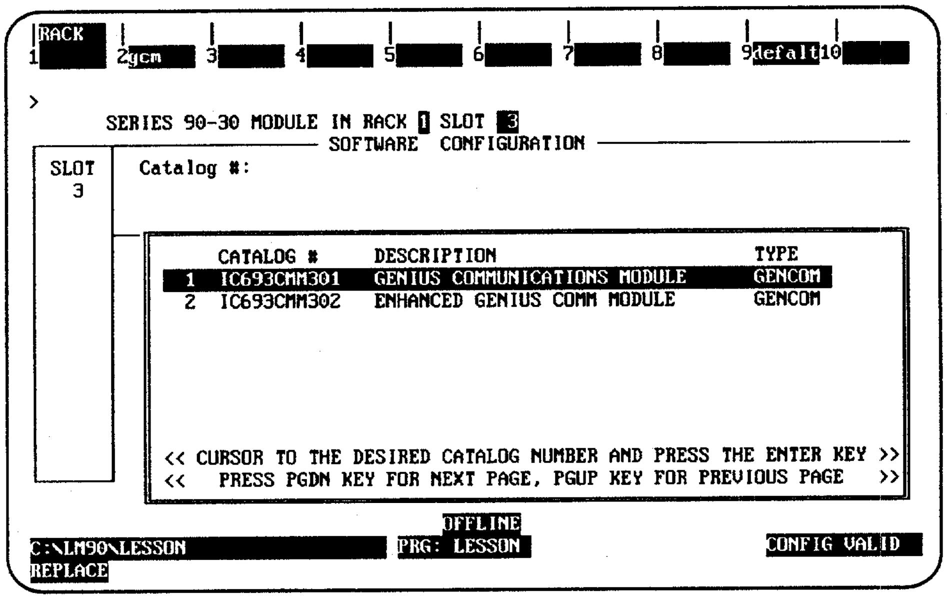
```
RACK
1pcm   2hsc   3frgn   4di   5apm   6   7   8   9   10
>
        SERIES 90-30 MODULE IN RACK 0 SLOT 9
                          SOFTWARE  CONFIGURATION
 SLOT    Catalog #:
  9
              CATALOG #      DESCRIPTION                      TYPE
            1  IC693ADC311   ALPHANUMERIC DISPLAY COPROC       ADC

         << CURSOR TO THE DESIRED CATALOG NUMBER AND PRESS THE ENTER KEY >>
         <<   PRESS PGDN KEY FOR NEXT PAGE, PGUP KEY FOR PREVIOUS PAGE    >>
                                               OFFLINE
C:\LM90\LESSON                               PRG: LESSON          CONFIG VALID
REPLACE
```

3. With the cursor positioned on IC693ADC311, press the **Enter** key.

Section 14: Configuring a GCM or Enhanced GCM

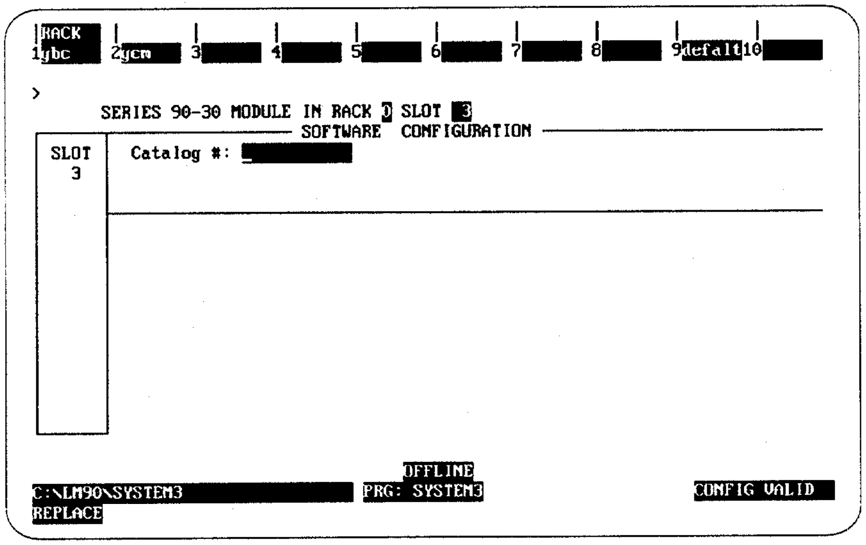
The Genius Communications Module provides the means for configuring global data to and from the Series 90-30 programmable controller. To configure the Genius Communications Module:

1. Move the cursor to the correct slot.
2. Press **Genius (F2)** and then **GCM (F2)** from the I/O Configuration Rack screen to display a list of catalog numbers and modules.



```
RACK  |     |     |     |     |     |     |     |       |
1     2gcm  3     4     5     6     7     8     9defalt10

>
      SERIES 90-30 MODULE IN RACK 0 SLOT 3
                          SOFTWARE CONFIGURATION
 SLOT   Catalog #:
  3

             CATALOG #      DESCRIPTION                       TYPE
          1  IC693CMM301    GENIUS COMMUNICATIONS MODULE      GENCOM
          2  IC693CMM302    ENHANCED GENIUS COMM MODULE       GENCOM

        << CURSOR TO THE DESIRED CATALOG NUMBER AND PRESS THE ENTER KEY >>
        <<   PRESS PGDN KEY FOR NEXT PAGE, PGUP KEY FOR PREVIOUS PAGE   >>

                                            OFFLINE
C:\LM90\LESSON                            PRG: LESSON           CONFIG VALID
REPLACE
```

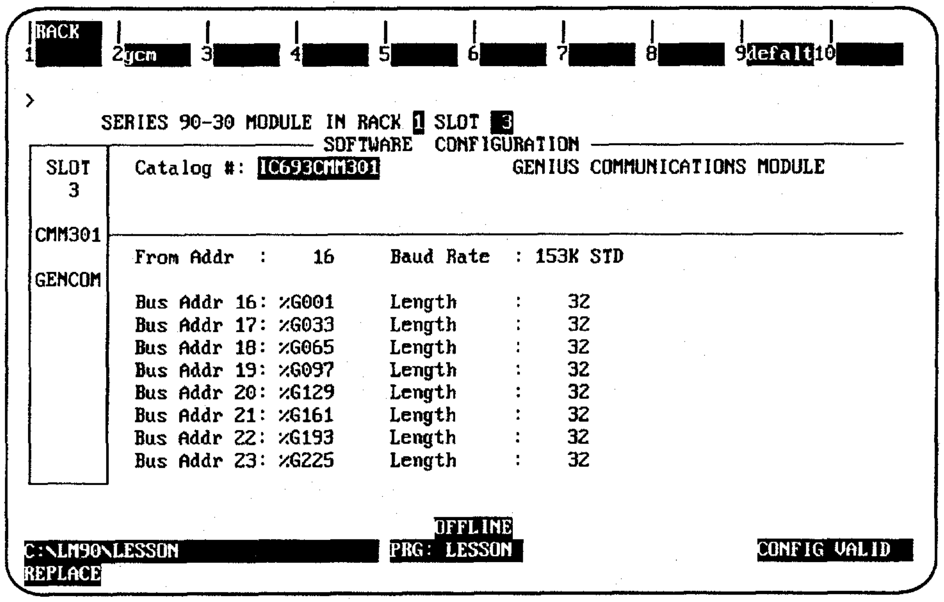
3. Then press **F2** (genius). The following screen will appear.



```
RACK  |     |     |     |     |     |     |     |       |
1gbc  2gcm  3     4     5     6     7     8     9defalt10

>
      SERIES 90-30 MODULE IN RACK 0 SLOT 3
                          SOFTWARE CONFIGURATION
 SLOT   Catalog #:
  3

                                            OFFLINE
C:\LM90\SYSTEM3                           PRG: SYSTEM3          CONFIG VALID
REPLACE
```

4. Press **F2** (gcm) and then press the **Enter** key to select it. The following screen will appear.

5. With the cursor positioned on IC693CMM301, press the **Enter** key to display the detail screen for the Genius Communications Module.



```
 RACK   |     |     |     |     |     |     |     |       |
1      2gcm  3     4     5     6     7     8     9defalt10

>
         SERIES 90-30 MODULE IN RACK 1 SLOT 3
 ________________________ SOFTWARE  CONFIGURATION __________________
| SLOT  | Catalog #: IC693CMM301          GENIUS COMMUNICATIONS MODULE
|   3   |
|       |
|CMM301 |____________________________________________________________
|       | From Addr   :    16       Baud Rate   : 153K STD
|GENCOM |
|       | Bus Addr 16: %G001        Length      :     32
|       | Bus Addr 17: %G033        Length      :     32
|       | Bus Addr 18: %G065        Length      :     32
|       | Bus Addr 19: %G097        Length      :     32
|       | Bus Addr 20: %G129        Length      :     32
|       | Bus Addr 21: %G161        Length      :     32
|       | Bus Addr 22: %G193        Length      :     32
|       | Bus Addr 23: %G225        Length      :     32
|_______|

                                     OFFLINE
C:\LM90\LESSON                      PRG: LESSON                   CONFIG VALID
REPLACE
```

6. Complete the detail screen, using the definitions provided in the following table, and press the **Enter** key again.

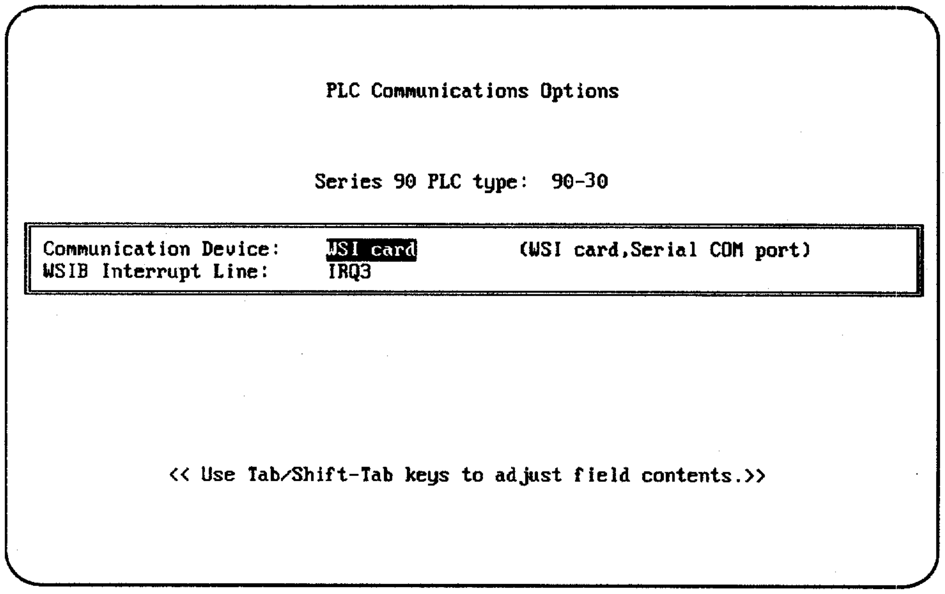
Field	Description
From Address	Specifies the serial bus address from where the outgoing global data is sent. The valid range for this field is **16** to **23**, inclusive. By specifying one bus address as output, the other seven serial bus addresses, by default, receive incoming global data.
Baud Rate	Selections for the baud rate include **153K STD**, **76.8K**, **38.4K**, and **153K EXT**
Length	The global references are fixed fields, referring to the address associated with each of the eight serial bus addresses **16** through **23**. Each Length field represents the amount of data transferred to or from the corresponding serial bus address, in units of bits. The length value, when added to the fixed starting address, must not exceed %G256, the maximum allowable global data reference.

The table below states the allowable values for each length, validated at the time of entry.

Bus Address	Starting Address	Valid Lengths
16	%G001	0 – 256
17	%G033	0 – 224
18	%G065	0 – 192
19	%G097	0 – 160
20	%G129	0 – 128
21	%G161	0 – 96
22	%G193	0 – 64
23	%G225	0 – 32

PLC Communications—
WSI Version with Non-Configurable Interrupt Request

In this example, again the WSI version of software is installed, but a non-configurable WSIB1 Board was used. The interrupt line IRQ3 is displayed in the "WSIB Interrupt Line" field and cannot be changed. Again, there are no options to select; the screen is presented only to provide information.



```
                         PLC Communications Options

                         Series 90 PLC type:  90-30

 Communication Device:      WSI card         (WSI card,Serial COM port)
 WSIB Interrupt Line:       IRQ3

              << Use Tab/Shift-Tab keys to adjust field contents.>>
```

Saving the Programmer Environment Setup

To save the setup information in a file for future use, first press **Esc** (the Escape key) to return to the "LOGICMASTER 90 Setup File Editor" screen. Then press **Save (F10)** from the Setup File Editor menu. After the file is saved, the software displays the message:

```
Setup File saved successfully as c:\lm90\lm90.prd.
```

Press any key to return to the Setup File Editor menu. Then, press the **Escape** key to return to the menu of Series 90 PLCs and functions.

Appendix D Instruction Mnemonics

In program display/edit mode, you can quickly enter or search for a programming instruction by typing the ampersand (**&**) character followed by the instruction's mnemonic. For some instructions, you can also specify a reference address or nickname, a label, or a location reference address.

This appendix lists the mnemonics of the programming instructions for Logicmaster 90-30/20/Micro programming software. At any time during programming, you can display a help screen with these mnemonics by pressing the **ALT** and **I** keys (i.e., hold down the **ALT** key and then tap the **I** key).

Function Group	Instruction	Mnemonic					
		All	INT	DINT	BIT	BYTE	WORD
Contacts	Any Contact	&CON	&CON				
	Normally Open Contact	&NOCON	&NOCON				
	Normally Closed Contact	&NCCON	&NCCON				
	Continuation Contact	&CONC	&CONC				
Coils	Any Coil	&COI	&COI				
	Normally Open Coil	&NOCOI	&NOCOI				
	Negated Coil	&NCCOI	&NCCOI				
	Positive Transition Coil	&PCOI	&PCOI				
	Negative Transition Coil	&NCOI	&NCOI				
	SET Coil	&SL	&SL				
	RESET Coil	&RL	&RL				
	Retentive SET Coil	&SM	&SM				
	Retentive RESET Coil	&RM	&RM				
	Retentive Coil	&NOM	&NOM				
	Negated Retentive Coil	&NCM	&NCM				
	Continuation Coil	&COILC	&COILC				
Links	Horizontal Link	&HO	&HO				
	Vertical Link	&VE	&VE				
Timers	On Delay Timer	&ON	&ON				
	Elapsed Timer	&TM	&TM				
	Off Delay Timer	&OF	&OF				
Counters	Up Counter	&UP	&UP				
	Down Counter	&DN	&DN				
Math	Addition	&AD	&AD_I	&AD_DI			
	Subtraction	&SUB	&SUB_I	&SUB_DI			
	Multiplication	&MUL	&MUL_I	&MUL_DI			
	Division	&DIV	&DIV_I	&DIV_DI			
	Modulo	&MOD	&MOD_I	&MOD_DI			
	Square Root	&SQ	&SQ_I	&SQ_DI			
Relational	Equal	&EQ	&EQ_I	&EQ_DI			
	Not Equal	&NE	&NE_I	&NE_DI			
	Greater Than	&GT	&GT_I	&GT_DI			
	Greater or Equal	&GE	&GE_I	&GE_DI			
	Less Than	&LT	&LT_I	&LT_DI			
	Less Than or Equal	&LE	&LE_I	&LE_DI			

Function Group	Instruction	Mnemonic					
		All	INT	DINT	BIT	BYTE	WORD
Bit Operation	AND	&AN					&AN_W
	OR	&OR					&OR_W
	Exclusive OR	&XO					&XO_W
	NOT	&NOT					&NOT_W
	Bit Shift Left	&SHL					&SHL_W
	Bit Shift Right	&SHR					&SHR_W
	Bit Rotate Left	&ROL					&ROL_W
	Bit Rotate Right	&ROR					&ROR_W
	Bit Test	&BT					&BT_W
	Bit Set	&BS					&BS_W
	Bit Clear	&BCL					&BCL_W
	Bit Position	&BP					&BP_W
	Masked Compare	&MCM					&MCM_W
Data Move	Move	&MOV	&MOV_I		&MOV_BI		&MOV_W
	Block Move	&BLKM	&BLKM_I				&BLKM_W
	Block Clear	&BLKC					
	Shift Register	&SHF			&SHF_BI		&AR_W
	Bit Sequencer	&BI					
	Communications Request	&COMMR					
Table	Array Move	&AR	&AR_I	&AR_DI	&AR_BI	&AR_BY	&MOV_W
	Search Equal	&SRCHE	&SRCHE_I	&SRCHE_DI		&SRCHE_BY	&BLKM_W
	Search Not Equal	&SRCHN	&SRCHN_I	&SRCHN_DI		&SRCHN_BY	&AR_W&SRC
	Search Greater Than	&SRCHGT	&SRCHGT_I	&SRCHGT_DI		&SRCHGT_BY	H_W&SRCHN
	Search Greater Than or Equal	&SRCHGE	&SRCHGE_I	&SRCHGE_DI		&SRCHGE_BY	_W&SRCHGT
	Search Less Than	&SRCHLT	&SRCHLT_I	&SRCHLT_DI		&SRCHLT_BY	_W&SRCHGE
	Search Less Than or Equal	&SRCHLE	&SRCHLE_I	&SRCHLE_DI		&SRCHLE_BY	_W&SRCHLT_
Conversion	Convert to Signed Integer	&I					
	Convert BCD–4 to Signed Integer	&I_BCD4					
		&TO_INT					
	Convert to BCD–4	&BCD4					
		&TO_BCD4					
Control	Call a Subroutine	&CA					
	Do I/O	&DO					
	PID – ISA Algorithm	&PIDIS					
	PID – IND Algorithm	&PIDIN					
	End	&END					
	Rung Explanation	&COMME					
	System Services Request	&SV					
	Master Control Relay	&MCR					
	End Master Control Relay	&ENDMCR					
	Nested Master Control Relay	&MCRN					
	Nested End Master Control Relay	&ENDMCRN					
	Jump	&JUMP	&JUMP				
	Nested Jump	&JUMPN	&JUMPN				
	Label	&LABEL	&LABEL				
	Nested Label	&LABELN	&LABELN				

Appendix E Key Functions

The following table lists the keyboard functions that are active in the Logicmaster 90-30/20/Micro software environment. This information may also be displayed on the programmer screen by pressing **ALT-K** to access key help.

Key Sequence	Description	Key Sequence	Description
Keys Available throughout the Software Package			
ALT-A	Abort.	CTRL-Break	Exit package.
ALT-C	Clear field.	Esc	Zoom out.
ALT-M	Change programmer mode.	CTRL-Home	Previous command line contents.
ALT-R	Change PLC **RUN/STOP** state.	CTRL-End	Next command line contents.
ALT-E	Toggle status area.	CTRL−←	Cursor left within the field.
ALT-J	Toggle the command line.	CTRL−→	Cursor right within the field.
ALT-L	List directory files.	CTRL-D	Decrement reference address.
ALT-P	Print screen.	CTRL-U	Increment reference address.
ALT-H	Help.	Tab	Change/increment field contents.
ALT-K	Key help.	Shift-Tab	Change/decrement field contents.
ALT-I	Instruction mnemonic help.	Enter	Accept field contents.
ALT-T	Start **TEACH** mode.	CTRL-E	Display last system error.
ALT-Q	Stop **TEACH** mode.	F12 (or Keypad −)	Toggle discrete reference.
ALT-n	Playback file n (n = 0 thru 9).	F11 (or Keypad *)	Override discrete reference.
Keys Available in the Program Editor Only			
ALT-B	Toggle text editor bell.	Keypad +	Accept rung.
ALT-D	Delete rung element.	Enter	Accept rung.
ALT-S	Store block to PLC and disk.	CTRL-PgUp	Previous rung.
ALT-X	Display zoom level.	CTRL-PgDn	Next rung.
ALT-N	Toggle nickname/reference address.	~	Horizontal shunt.
ALT-U	Update disk.	|	Vertical shunt.
ALT-V	Variable table window.	Tab	Go to the next operand field.
Special Keys			
ALT-O	Password override. Available only on the Password screen in the configuration software.		

The Help card on the next page contains a listing of the key help and also the instruction mnemonics help text for Logicmaster 90 software. This card is printed in triplicate and is perforated for easier removal from the manual.

If one field is increased beyond the default of 32 bits, the next field(s) must be decreased to avoid overlapping addresses. For example, if 64 bits is specified for serial bus address 16, the data transmitted from this node maps to %G001 to %G064. Therefore, a length of 0 must be assigned to serial bus address 17. The *Length* fields will be automatically adjusted so that overlapping %G addresses are not possible.

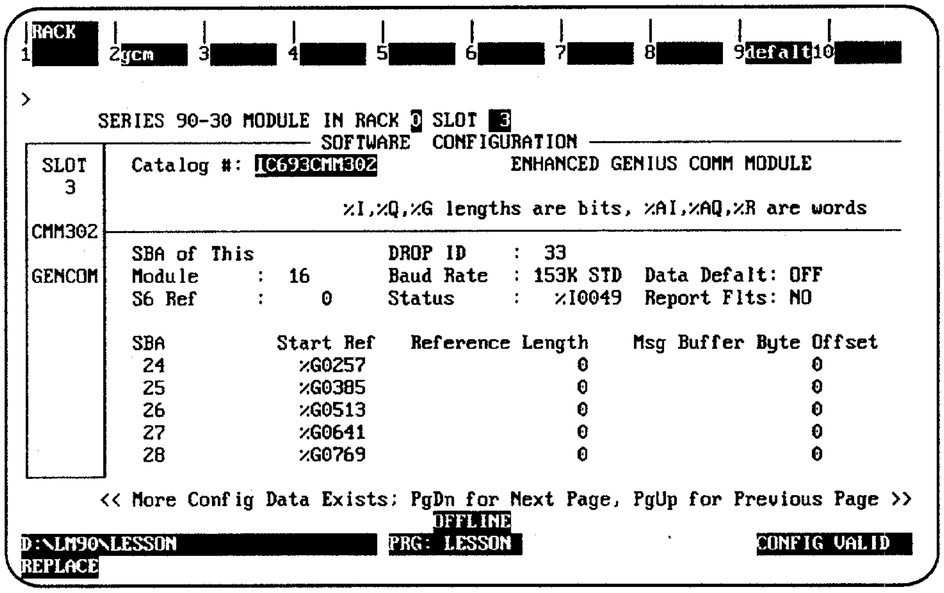
Enhanced Genius Communications Module

Note

Beginning with Release 4.5, the I/O configuration is no longer limited to two Enhanced Genius Communications Modules. The number of Enhanced GCM modules that can be configured is now limited only by the following:

- The presence of a Genius Communications Module. If a GCM module is already configured, an Enhanced GCM module cannot be configured.
- The maximum configuration size of the PLC. If the references assigned to the Enhanced GCM Modules causes the I/O configuration size to exceed the maximum allowed, then the configuration cannot be stored to the PLC.

1. To configure an Enhanced Genius Communications Module, position the cursor on IC693CMM302 in the catalog list of modules, and press the **`Enter`** key. The following detail screen will be displayed.



```
 RACK
1      2gcm  3     4     5     6     7     8     9defalt10
>
      SERIES 90-30 MODULE IN RACK 0 SLOT 3
                            SOFTWARE  CONFIGURATION
 SLOT   Catalog #: IC693CMM302            ENHANCED GENIUS COMM MODULE
  3
                        %I,%Q,%G lengths are bits, %AI,%AQ,%R are words
CMM302
        SBA of This          DROP ID    :  33
GENCOM  Module     :  16     Baud Rate  : 153K STD  Data Defalt: OFF
        S6 Ref     :    0    Status     :   %I0049  Report Flts: NO

        SBA          Start Ref    Reference Length    Msg Buffer Byte Offset
         24          %G0257                   0                          0
         25          %G0385                   0                          0
         26          %G0513                   0                          0
         27          %G0641                   0                          0
         28          %G0769                   0                          0

      << More Config Data Exists; PgDn for Next Page, PgUp for Previous Page >>
                                      OFFLINE
D:\LM90\LESSON                   PRG: LESSON                     CONFIG VALID
REPLACE
```

2. Complete the detail screen, using the definitions provided in the following table. Then, press the **Enter** key again.

Parameter	Description
Serial Bus Address (SBA) of this Module	Indicates the Serial Bus Address of this module. This is the address from which global data will be sent. Values are **0** to **31**. (Default = **16**)
Baud Rate	Selections for the baud rate include **153K STD**, **76.8K**, **38.4K**, and **153K EXT** (Default = **153K STD**)
Data Default	If set to **OFF**, the Enhanced GCM module will either zero out data being supplied to the host PLC whenever the corresponding Genius device supplying the data ceases to communicate on the bus or zero out data being transmitted onto the bus when its host PLC ceases to scan its own I/O. If set to **HOLD**, the Enhanced GCM module will continue to pass on the last valid state sent to it in either of these situations. (Default = **OFF**)
Series Six Reference	Specifies the register location in a Series Six or Series Five CPU that should be reserved for the global data that will be transmitted to it by the Enhanced GCM module. A value of zero indicates that no register location should be reserved. Values are **0** – **16,383**, inclusive. (Default = **0**)
Status	Specifies the location in PLC memory used to hold the status specified by the Enhanced GCM module. The memory type for this field is restricted to %I. (Default = the next highest available %I value)
Report Faults	If set to **YES,** the Enhanced GCM module will accept fault mail from its host Series 90-30 PLC and transmit it onto the bus as a Genius datagram which can be interpreted by a Series 90-70 Genius Bus Controller. If **NO** is selected, the Enhanced GCM module will not pass on any fault information. (Default = **NO**)
Drop ID	This parameter is used by a Series 90-70 PLC to identify the location of a fault that has been reported by a Series 90-30 PLC via an Enhanced GCM module. The Report Faults option of the Enhanced GCM module must also be enabled. When locating two Enhanced GCM modules in one Series 90-30 PLC, it is recommended that you assign a ***unique*** Drop ID to each module in order to avoid conflicting identification information if both modules are set up to report faults. Valid range is **16** through **254**. (Default = **33**)

The following parameters apply for each configurable Serial Bus Address (SBAs 0 through 31):

Parameter	Description
Start Reference	The starting reference address from or to which global data is being transferred. The reference type can be %G, %R, %AI, %AQ, %I, or %Q.
Reference Length	The total length of the reference data, starting at the address specified by START REF. Values are 0 to 128 bytes; however, the units for this field will be bits if the configured reference type is %I, %Q, or %G, and words if the configured reference type is %AI, %AQ, or %R.
Message Buffer Byte Offset	The offset, measured in bytes, from the start of an incoming message at which the Enhanced GCM module will start to extract data from the incoming message. The first byte extracted is placed in PLC memory at the start reference given in the configuration of the Serial Bus Address of the device transmitting the message. A value of zero indicates that the first byte of the incoming message is placed at the starting references defined for each Serial Bus Address. The offset for the Serial Bus Address of the Enhanced GCM module itself is always zero; it cannot be changed. (Default = 0) For each Serial Bus Address, the sum of the reference length and the message buffer byte offset cannot exceed 128 bytes. Since the Serial Bus Address of this module is used to transmit global data, its reference address may overlap with other configured reference addresses.

3. The Default (F9) can be used to set the reference addresses and reference lengths to a combined Scheme 1 and Scheme 2 Genius global data. Serial Bus Addresses 16 through 23 may be configured to use Genius Scheme 1, and Serial Bus Addresses 24 through 31 may be configured to use Genius Scheme 2. All other Serial Bus Addresses are set to their default value. In addition, all message buffer byte offsets are set to zero.

For additional information on the configuration of devices and remote drops, see:

Genius Discrete and Analog Blocks User's Manual (GEK-90486-2) – includes instructions for configuring most I/O blocks.

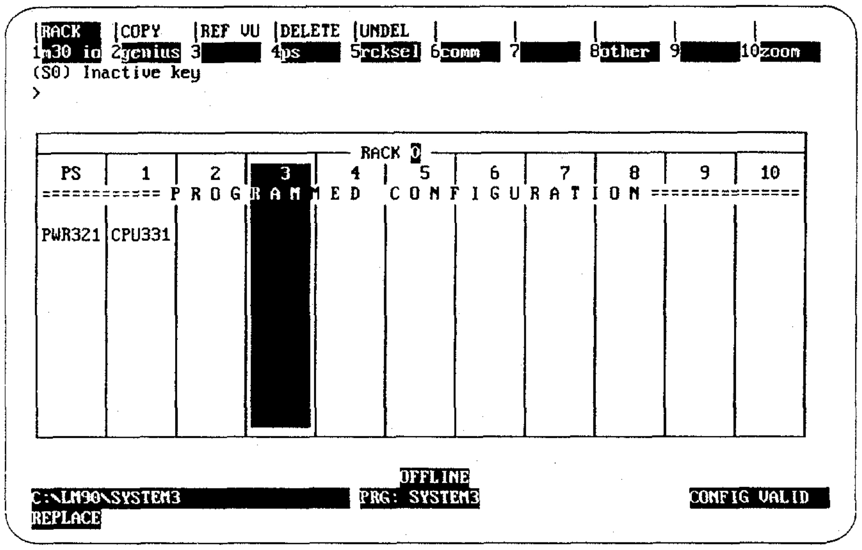
Genius I/O System User's Manual (GEK-90486-1) – details the data that can be transferred using Read Configuration and Write Configuration COMREQs.

Section 15: Configuring a Genius Bus Controller

The Genius Bus Controller (GBC) must be configured as part of the Series 90-30 PLC system using the Logicmaster 90-30/20/Micro software (release 5 or later) or a Hand-held Programmer (HHP). This includes configuring parameters for the GBC module itself, and parameters specific to the devices on the GBC's bus.

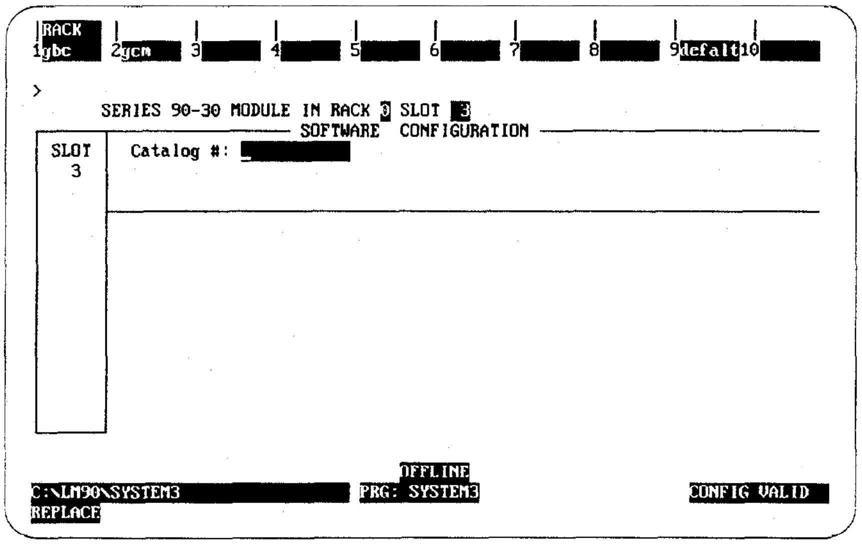
The LM90 configurator software program (release 5 or later) can be used to configure the module in the offline mode. Once the complete set of configuration data has been entered, it must then be downloaded to the PLC (in the online mode) to become effective in the Genius Bus Controller.

1. The GBC is configured by completing setup screens in the Logicmaster 90-30 configuration software. The setup screens that are used for this module are shown and described below. In the I/O configuration screen, place the cursor at the slot representation corresponding to the GBC's installed location in the PLC rack.



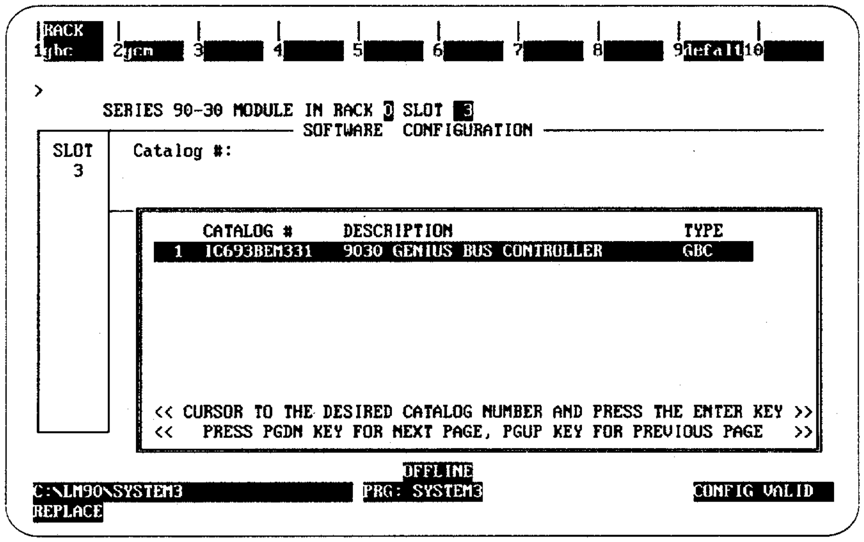
```
|RACK    |COPY    |REF VU  |DELETE  |UNDEL   |        |        |        |        |
1 30 io  2genius  3        4ps      5rcksel  6comm    7        8other   9        10zoom
(S0) Inactive key
>

                               RACK 0
  PS   |  1   |  2   |  3   |  4   |  5   |  6   |  7   |  8   |  9   |  10
============ P R O G R A M M E D   C O N F I G U R A T I O N ==============
PWR321 |CPU331|

                                    OFFLINE
C:\LM90\SYSTEM3                     PRG: SYSTEM3                CONFIG VALID
REPLACE
```

2. Then press **F2** (genius). The following screen will appear.



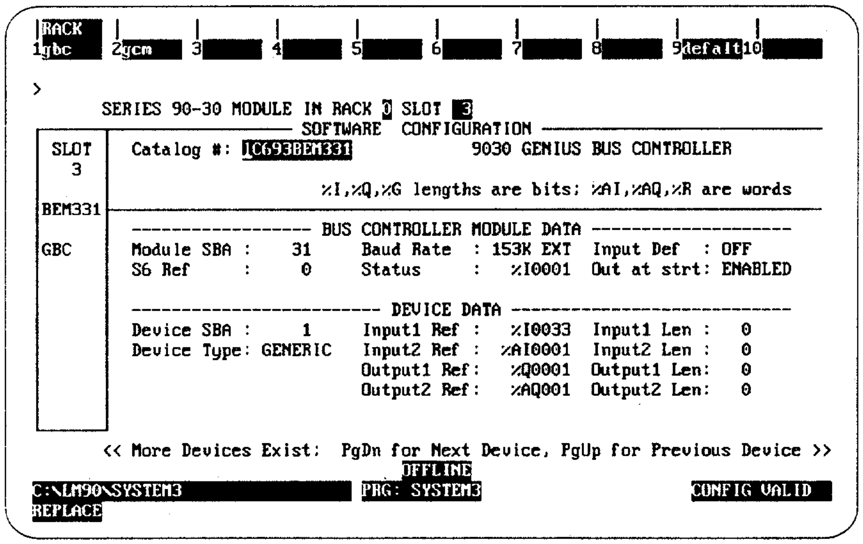
```
|RACK    |        |        |        |        |        |        |        |        |
1gbc     2gcm     3        4        5        6        7        8        9defalt  10

>
          SERIES 90-30 MODULE IN RACK 0 SLOT 3
                         SOFTWARE  CONFIGURATION
 SLOT  |  Catalog #:
   3   |

                                    OFFLINE
C:\LM90\SYSTEM3                     PRG: SYSTEM3                CONFIG VALID
REPLACE
```

3. Press **F1** (gbc) and then press the **Enter** key to select the GBC. The following screen will appear. (Note that the "**defalt**" softkey, **F9**, is inactive.)



```
 RACK
1gbc   2gcm   3      4      5      6      7      8      9defalt10
>
      SERIES 90-30 MODULE IN RACK 0 SLOT 3
                              SOFTWARE  CONFIGURATION
 SLOT   Catalog #:
  3

           CATALOG #     DESCRIPTION                      TYPE
        1  IC693BEM331   9030 GENIUS BUS CONTROLLER       GBC




     << CURSOR TO THE DESIRED CATALOG NUMBER AND PRESS THE ENTER KEY >>
     <<   PRESS PGDN KEY FOR NEXT PAGE, PGUP KEY FOR PREVIOUS PAGE   >>
                                       OFFLINE
C:\LM90\SYSTEM3                       PRG: SYSTEM3           CONFIG VALID
REPLACE
```

4. Press the **Enter** key to select the GBC. Complete the GBC configuration entries in the following screen:



```
 RACK
1gbc   2gcm   3      4      5      6      7      8      9defalt10
>
      SERIES 90-30 MODULE IN RACK 0 SLOT 3
                              SOFTWARE  CONFIGURATION
 SLOT   Catalog #: IC693BEM331          9030 GENIUS BUS CONTROLLER
  3
                     %I,%Q,%G lengths are bits; %AI,%AQ,%R are words
BEM331
        ------------------ BUS CONTROLLER MODULE DATA --------------------
GBC     Module SBA :    31     Baud Rate  : 153K EXT  Input Def   : OFF
        S6 Ref     :    0      Status     :   %I0001  Out at strt: ENABLED

        ------------------------ DEVICE DATA ----------------------------
        Device SBA :    1      Input1 Ref :   %I0033  Input1 Len :   0
        Device Type: GENERIC   Input2 Ref :  %AI0001  Input2 Len :   0
                               Output1 Ref:   %Q0001  Output1 Len:   0
                               Output2 Ref:   %AQ001  Output2 Len:   0

      << More Devices Exist:  PgDn for Next Device, PgUp for Previous Device >>
                                       OFFLINE
C:\LM90\SYSTEM3                       PRG: SYSTEM3           CONFIG VALID
REPLACE
```

Note that the configuration screen consists of two parts: module-specific data (BUS CONTROLLER MODULE DATA) and device-specific data (DEVICE DATA).

The default entries can be used as is, or changed. *Until a valid configuration is stored to the PLC CPU, the GBC will not operate on the bus, and its Channel OK LED will not light.*

For additional information on the configuration of devices and remote drops, see:

Genius Discrete and Analog Blocks User's Manual (GEK-90486-2) – includes instructions for configuring most I/O blocks.

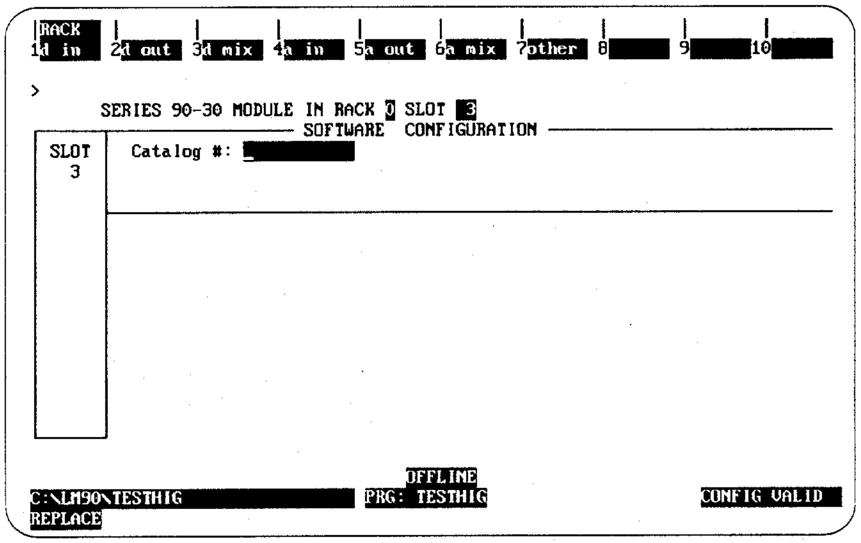
Genius I/O System User's Manual (GEK-90486-1) – details the data that can be transferred using Read Configuration and Write Configuration COMREQs.

Series 90-30 Genius Bus Controller User's Manual (GFK-1034) – includes configuration and additional details on relevant several topics including configuring for global data, as well as details on assigned configuration parameters.

Section 16: Configuring a High Density Analog Output Module

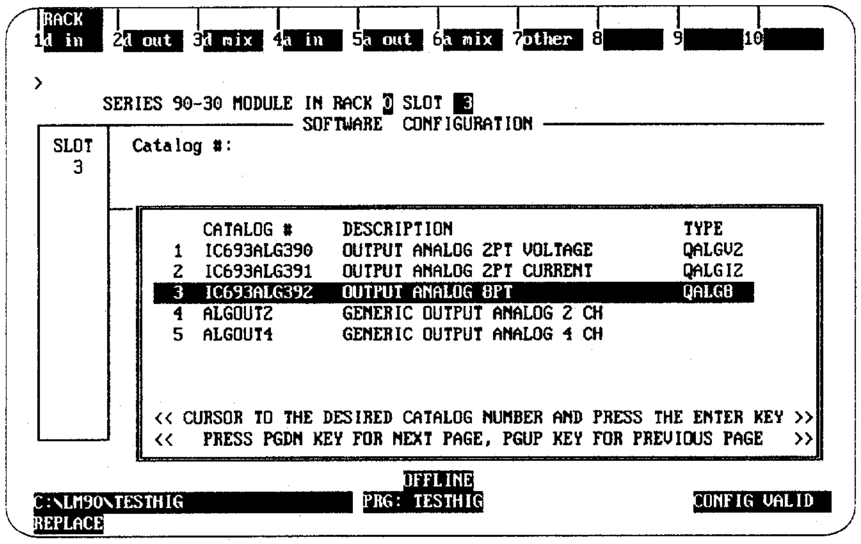
To configure a High Density Analog Output module, follow these steps:

1. Move the cursor to the desired rack and slot location. The slot may be either unconfigured or previously configured.
2. Press the **lm30 io** key (**F1**). You will then see a screen similar to the following:



```
 RACK  |      |      |      |      |      |        |    |    |
1d in  2d out 3d mix 4a in  5a out 6a mix 7other   8    9    10

>
      SERIES 90-30 MODULE IN RACK 0 SLOT 3
                             SOFTWARE  CONFIGURATION
 SLOT    Catalog #:
   3

C:\LM90\TESTHIG                    OFFLINE
REPLACE                            PRG: TESTHIG           CONFIG VALID
```

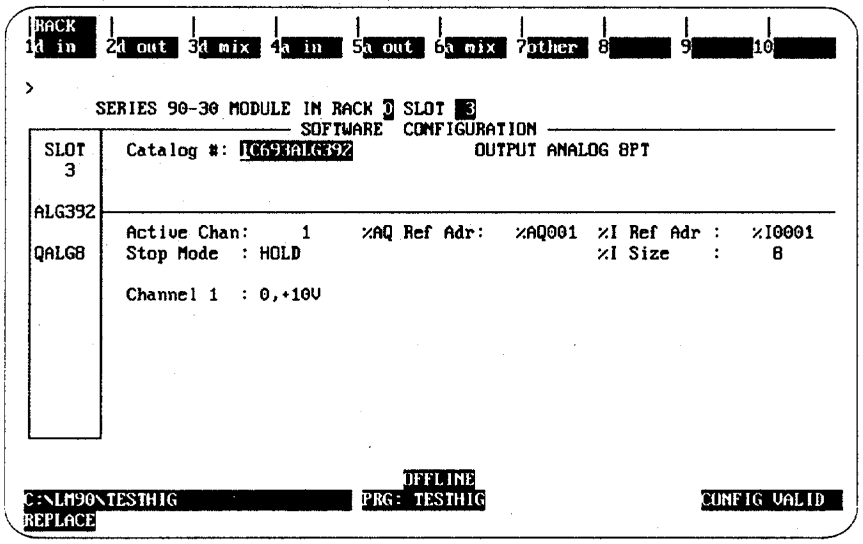
3. Press the **a out** key (**F5**). Your screen will now look like the one displayed below:



```
 RACK  |      |      |      |      |      |        |    |    |
1d in  2d out 3d mix 4a in  5a out 6a mix 7other   8    9    10

>
      SERIES 90-30 MODULE IN RACK 0 SLOT 3
                             SOFTWARE  CONFIGURATION
 SLOT    Catalog #:
   3
            CATALOG #     DESCRIPTION                   TYPE
          1 IC693ALG390   OUTPUT ANALOG 2PT VOLTAGE     QALGV2
          2 IC693ALG391   OUTPUT ANALOG 2PT CURRENT     QALGI2
          3 IC693ALG392   OUTPUT ANALOG 8PT             QALG8
          4 ALGOUT2       GENERIC OUTPUT ANALOG 2 CH
          5 ALGOUT4       GENERIC OUTPUT ANALOG 4 CH

       << CURSOR TO THE DESIRED CATALOG NUMBER AND PRESS THE ENTER KEY >>
       <<   PRESS PGDN KEY FOR NEXT PAGE, PGUP KEY FOR PREVIOUS PAGE   >>

C:\LM90\TESTHIG                    OFFLINE
REPLACE                            PRG: TESTHIG           CONFIG VALID
```

4. Move the cursor to the IC69ALG392 selection as shown above. Then press **Enter**.

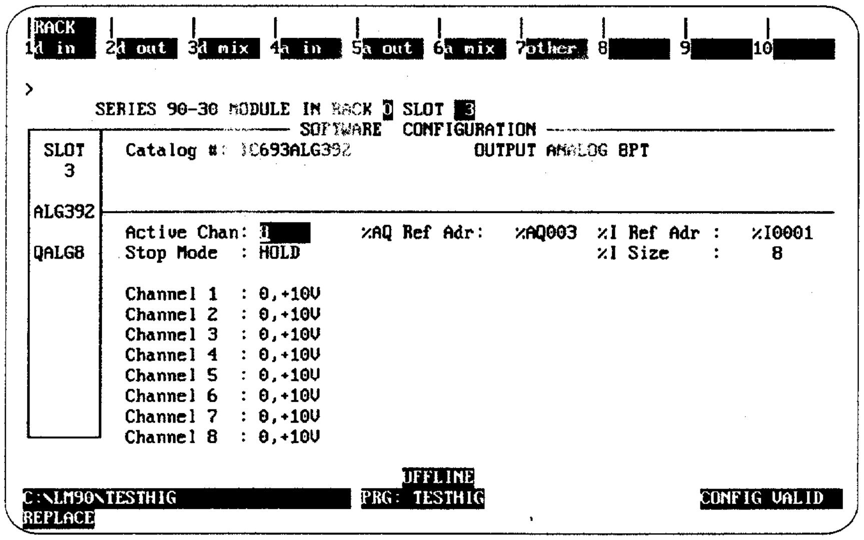
The next screen that appears will look like the one displayed below:



```
|RACK  |      |      |      |      |      |      |      |      |
1I in  2I out 3I mix 4A in  5A out 6A mix 7other 8      9      10

>
         SERIES 90-30 MODULE IN RACK 0 SLOT 3
 ________________________________ SOFTWARE  CONFIGURATION ____________________
| SLOT |  Catalog #: IC693ALG392            OUTPUT ANALOG 8PT
|  3   |
|      |
|ALG392|______________________________________________________________________
|      |  Active Chan:     1     %AQ Ref Adr:   %AQ001  %I Ref Adr :   %I0001
|QALG8 |  Stop Mode  : HOLD                             %I Size    :     8
|      |
|      |  Channel 1  : 0,+10V
|      |
|______|

                                        OFFLINE
C:\LM90\TESTHIG                        PRG: TESTHIG                 CONFIG VALID
REPLACE
```

5. Enter the remaining configuration parameters on this screen. You can move your cursor from field to field by pressing the **Cursor Movement** (or **Arrow**) keys. When you are in the field you want to modify, you can either type in your choice or press the **Tab** key to scroll through the available selections (or **Shift-Tab** to reverse the direction of the scrolling).

   The default number of Active Channels (**Active Chan:**) is 1. You will not be able to configure additional channels until you change this field (by typing in the correct number (1 through 8) or by pressing the **Tab** key to increment the number). The screen displayed below shows the default selections after changing the **Active Chan:** field.



```
|RACK  |      |      |      |      |      |      |      |      |
1I in  2I out 3I mix 4A in  5A out 6A mix 7other 8      9      10

>
         SERIES 90-30 MODULE IN RACK 0 SLOT 3
 ________________________________ SOFTWARE  CONFIGURATION ____________________
| SLOT |  Catalog #: IC693ALG392            OUTPUT ANALOG 8PT
|  3   |
|      |
|ALG392|______________________________________________________________________
|      |  Active Chan: 8         %AQ Ref Adr:   %AQ003  %I Ref Adr :   %I0001
|QALG8 |  Stop Mode  : HOLD                             %I Size    :     8
|      |
|      |  Channel 1  : 0,+10V
|      |  Channel 2  : 0,+10V
|      |  Channel 3  : 0,+10V
|      |  Channel 4  : 0,+10V
|      |  Channel 5  : 0,+10V
|      |  Channel 6  : 0,+10V
|      |  Channel 7  : 0,+10V
|______|  Channel 8  : 0,+10V

                                        OFFLINE
C:\LM90\TESTHIG                        PRG: TESTHIG                 CONFIG VALID
REPLACE
```

Note

The entry in the **Stop Mode** field (**HOLD** or **DEFLOW**–which stands for default low) determines how the outputs will behave when the module is toggled from **RUN** to **STOP** mode. When this value is set to **HOLD** (the default), the outputs will retain their last state. When you change this value to **DEFLOW**, the output will go to zero.

Other Configuration Considerations

Channels are scanned in sequential, contiguous order with channel 1 being the first to be scanned.

The entry in **%I Size** will only accept 8 or 16. This field denotes the number of bits returned to the user.

The only allowable entries for the **%AQ Ref Adr** are %AQ addresses. The only allowable entries for the **%I Ref Adr** are %I addresses.

The following tables show the configuration parameters applicable to this module.

Parameter	Parameter Description	Parameter Values	Parameter Defaults	Parameter Units
Stop Mode	Hold last state from RUN to STOP	HOLD or DEFLOW	HOLD	N/A
ACTIVE Channels	Number of channels converted	1 through 8	1	Channels
%AQ Ref Adr	Starting address for the %AQ reference type	Standard range	%AQ0001, or next highest available	N/A
%I Ref Adr	Starting address for the %I reference type	Standard range	%I0001, or next highest available	N/A
%I Size	Number of %I status locations	8 or 16	8	bits
Range of channel output (not identified as such on the screen)	Type of output range	0, +10V; −10, +10V; 4,20mA, 0,20mA	0, +10V	Volts (Volt) mA (Curr)

Notes:

1. The **%AQ Ref Adr** field is the reference address for the %AQ data and points to the start of the locations in the %AQ memory where the output data to the module begins. Each channel provides 16 bits of analog output data as an integer value from 0 to 32,767 or −32,768 to 32,767. depending on the range type selected. For detailed information of the data format, see the "CPU Interface to Analog Modules" section in chapter 3 of the *Series 90-30 Programmable Controller I/O Module Specifications* (GFK-0898).

2. The **%I Ref Adr** is the reference address for the %I data and points to the start of the locations in the %I memory (i.e., the Input Table) where status information from the module is reported. The user can select the number of %I status locations reported to the PLC by editing the value in the **%I Size** field. Values allowable in the %I Size field are 8 or 16, which refer to the number of %I locations reported to the PLC.

The `%I Ref Adr` field will only accept %I for %I Size values 8 or greater; the data brought back is in the format that follows:

The first eight %I locations (available for %I SIZE values 8, 16)

%I Locations	Description
%I	Module OK – '0' indicates NOT OK, '1' indicates module OK
%I+1	User Supply OK – Indicates when user supply is in specified limits; reads a '0' when User supply below specified limit, '1' when User supply OK
%I+2 – %I+7	Reserved for future modules. Not used in this module.

Second eight locations – (available for %I SIZE values 16)

%I Locations	Description
%I+8	Channel #1 BROKEN WIRE – 0 = OK, 1 = Wire broken (I modes only)
%I+9	Channel #2 BROKEN WIRE – 0 = OK, 1 = Wire broken (I modes only)
%I+10	Channel #3 BROKEN WIRE – 0 = OK, 1 = Wire broken (I modes only)
%I+11	Channel #4 BROKEN WIRE – 0 = OK, 1 = Wire broken (I modes only)
%I+12	Channel #5 BROKEN WIRE – 0 = OK, 1 = Wire broken (I modes only)
%I+13	Channel #6 BROKEN WIRE – 0 = OK, 1 = Wire broken (I modes only)
%I+14	Channel #7 BROKEN WIRE – 0 = OK, 1 = Wire broken (I modes only)
%I+15	Channel #8 BROKEN WIRE – 0 = OK, 1 = Wire broken (I modes only)

One of four output ranges can be selected. Two of which are voltage ranges. The default range is 0–10V, where output voltage values ranging from 0 to 10 volts correspond to 0 to 32000 integer values from the 90-30 CPU. The –10 to +10V range, when selected, corresponds from –32000 to 32000 from the CPU over an output voltage range of –10 to +10V. The two current ranges are 4 to 20 mA, and 0 to 20 mA. In each of the current ranges values between 0 and 32000 are sent to the module. Depending on which range is selected, will determine if the module is in Current or Voltage mode.

The following tables shows values sent from the CPU to the module:

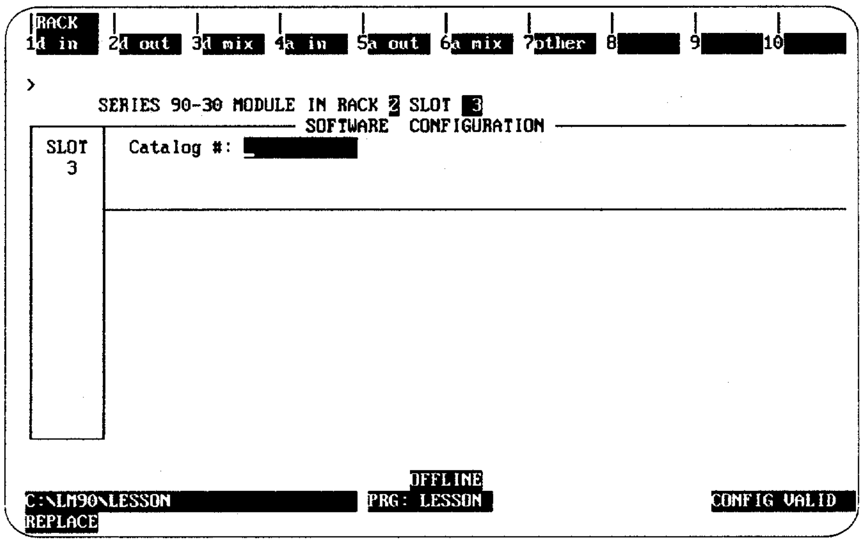
Range	Module Mode	*Allowed Values	Sent values from CPU
0 to 10 V	Voltage	0 to 32767	0 to 32767
−10 to 10 V	Voltage	− 32768 to 32767	−32768 to 32767
4 to 20 mA	Current	0 to 32000*	0 to 32767
0 to 20 mA	Current	0 to 32767	0 to 32767

*The phrase, *Allowed Values*, refers to the values that are valid. If you send a value > 32000, the module will truncate that value to 32000 before sending it to the DAC.

Section 17: Configuring an Analog Combo Module

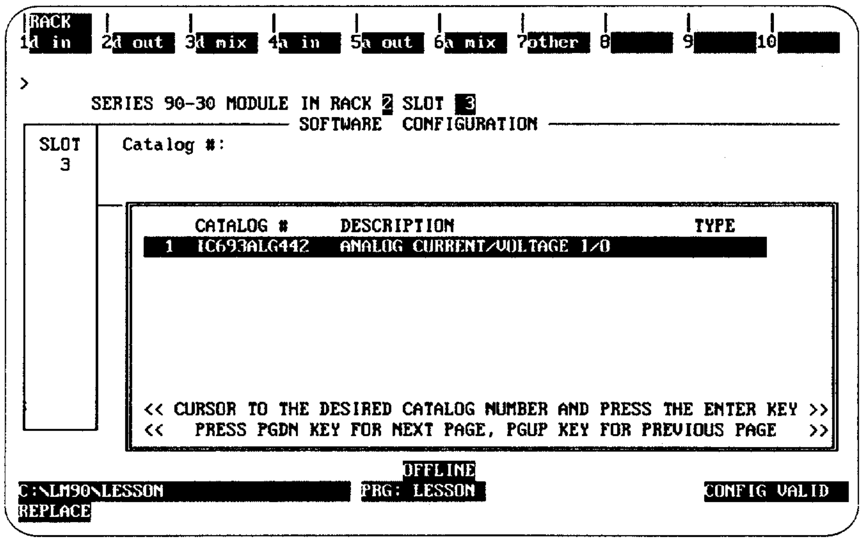
To configure an Analog Combo module, follow these steps:

1. Move the cursor to the desired rack and slot location. The slot may be either unconfigured or previously configured.
2. Press the **`lm30 io`** key (**`F1`**). Your screen will change to one similar to the one shown below.



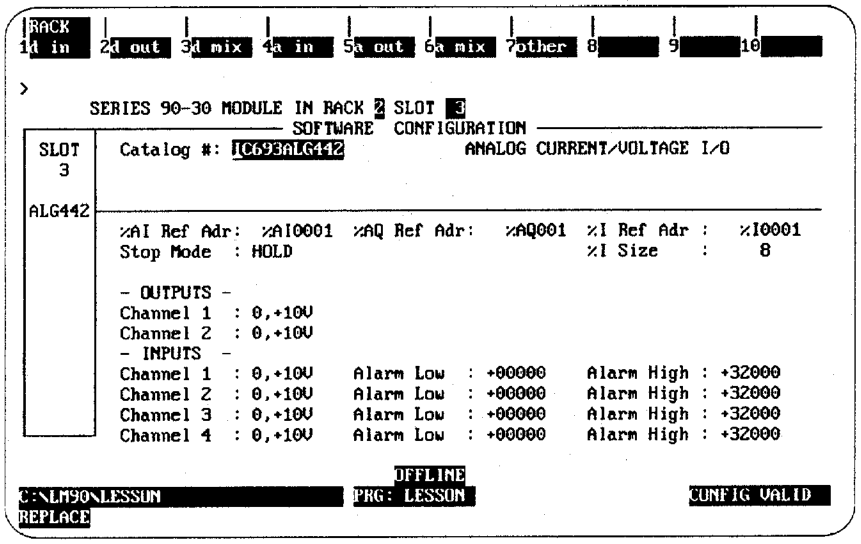
```
 RACK
1d in  2d out  3d mix  4a in  5a out  6a mix  7other  8      9      10
>
      SERIES 90-30 MODULE IN RACK 2 SLOT 3
                           SOFTWARE  CONFIGURATION
 SLOT   Catalog #: _
   3

                                              OFFLINE
C:\LM90\LESSON                               PRG: LESSON            CONFIG VALID
REPLACE
```

3. From this screen, press the **`a mix`** key (**`F6`**). Your screen will change to one similar to the one shown below.



```
 RACK
1d in  2d out  3d mix  4a in  5a out  6a mix  7other  8      9      10
>
      SERIES 90-30 MODULE IN RACK 2 SLOT 3
                           SOFTWARE  CONFIGURATION
 SLOT   Catalog #:
   3
            CATALOG #      DESCRIPTION                     TYPE
         1  IC693ALG442    ANALOG CURRENT/VOLTAGE I/O

      << CURSOR TO THE DESIRED CATALOG NUMBER AND PRESS THE ENTER KEY >>
      <<   PRESS PGDN KEY FOR NEXT PAGE, PGUP KEY FOR PREVIOUS PAGE   >>
                                              OFFLINE
C:\LM90\LESSON                               PRG: LESSON            CONFIG VALID
REPLACE
```

4. Currently, there is only one selection. (If more than one selection appears, use your **`Cursor Movement`** (or **`Arrow`**) keys to move to Catalog # IC693ALG442.) Press **`Enter`** to accept this selection and to move to the screen shown on the top of the next page.



```
|RACK   |       |       |       |       |       |       |       |       |
1d in  2d out  3d mix  4a in  5a out  6a mix  7other  8      9      10
>
      SERIES 90-30 MODULE IN RACK 0 SLOT 3
 ---------------------- SOFTWARE  CONFIGURATION ----------------------------
 SLOT  | Catalog #: IC693ALG442           ANALOG CURRENT/VOLTAGE I/O
  3    |
       |
ALG442 |-------------------------------------------------------------------
       | %AI Ref Adr:  %AI0001  %AQ Ref Adr:   %AQ001  %I Ref Adr :   %I0001
       | Stop Mode  : HOLD                             %I Size    :     8
       |
       | - OUTPUTS -
       | Channel 1  : 0,+10V
       | Channel 2  : 0,+10V
       | - INPUTS  -
       | Channel 1  : 0,+10V   Alarm Low  : +00000    Alarm High : +32000
       | Channel 2  : 0,+10V   Alarm Low  : +00000    Alarm High : +32000
       | Channel 3  : 0,+10V   Alarm Low  : +00000    Alarm High : +32000
       | Channel 4  : 0,+10V   Alarm Low  : +00000    Alarm High : +32000

                                     OFFLINE
C:\LM90\LESSON                      PRG: LESSON                 CONFIG VALID
REPLACE
```

5. All the remaining configuration need to be done on this screen. You can move your cursor from field to field by pressing the **Cursor Movement** (or **Arrow**) keys. When you are in the field you want to modify, you can either type in your choice or press the **Tab** key to scroll through the available selections (or **Shift-Tab** to reverse the direction of the scrolling).

Note

The entry in the **Stop Mode** field (**HOLD** or **DEFLOW**) determines how the outputs will behave when the module is toggled from **RUN** to **STOP** mode. When this value is set to **HOLD** (the default), the outputs will retain their last state. When you change this value to **DEFLOW**, the output will go to zero.

Other Configuration Considerations

The **Alarm Low** limit for each channel must be less than its corresponding **Alarm High** limit.

The entry in **%I Size** will only accept 8, 16 and 24. This field denotes the number of bits returned to the user.

The only allowable entries for the **%AQ Ref Adr** are %AQ addresses. The only allowable entries for the **%I Ref Adr** are %I addresses.

Analog Combo Configuration Parameters

Parameter	Parameter Description	Parameter Values	Parameter Defaults	Parameter Units
STOP MODE	Hold last state from RUN to STOP	HOLD or DEFLOW	HOLD	N/A
%AI ADR	Starting address for the %AI reference type	standard range	%AI0001, or next highest available	N/A
%AQ ADR	Starting address for the %AQ reference type.	standard range	%AQ0001, or next highest available	N/A
%I ADR	Starting address for the %I reference type	standard range	%I00001, or next highest available	N/A
%I SIZE	Number of %I status locations	8, 16, 24	8	bits
RANGE	Type of input range and range	0,+10 V, −10,+10 V, 4,20 mA, 0, 20mA	0,+10 V	volts (Volt) mA (Curr)
ALARM LO	Low limit alarm value	−32768 – 32752	0	User counts
ALARM HIGH	High limit alarm value	−32760 – 32760	+32000	User counts

Notes:

1. The **%AI Ref Adr** field is the reference address for the %AQ data and points to the start of the locations in the %AI memory where the output data to the module begins. Each channel provides 16 bits of analog output data as an integer value from 0 to 32,767 or −32,768 to 32,767. depending on the range type selected. For detailed information of the data format, see the "CPU Interface to Analog Modules" section in chapter 3 of the *Series 90-30 Programmable Controller I/O Module Specifications* (GFK-0898).

2. The **%AQ Ref Adr** field is the reference address for the %AQ data and points to the start of the locations in the %AQ memory where the output data to the module begins. Each channel provides 16 bits of analog output data as an integer value from 0 to 32,767 or −32,768 to 32,767. depending on the range type selected. For detailed information of the data format, see the "CPU Interface to Analog Modules" section in chapter 3 of the *Series 90-30 Programmable Controller I/O Module Specifications* (GFK-0898).

3. The **%I Ref Adr** is the reference address for the %I data and points to the start of the locations in the %I memory (i.e., the Input Table) where status information from the module is reported. You can select the number of %I status locations reported to the PLC by editing the value in the **%I Size** field. Values allowable in the %I Size field are 8 or 16, which refer to the number of %I locations reported to the PLC. For %I SIZE values 8 or greater, the data brought back is in the format described in the table on the next page.

First eight %I locations – (available for %I SIZE values 8, 16, 24)

%I Locations	Description
%I	Module OK – '0' indicates NOT OK, '1' indicates module OK
%I+1	User Supply OK – Indicates when user supply is in specified limits; reads a '0' when User supply below specified limit, '1' when User supply OK
%I+2 – %I+3	Reserved for future modules. Not used in this module.
%I + 4 – %I + 7	E2 COMMREQ Status Bits %I + 4 = 0 OK, if 1 = Invalid channel requested %I = 5 = 0 OK, if 1 = Invalid alarm setting Low > High or Low or High < 0 in unipolar mode

Second eight locations (available for %I SIZE values 16, 24)

%I Locations	Description
%I+8	Input: Ch #1 ALARM LO – '0' indicates value above limit, '1' below
%I+9	Input Ch #1 ALARM HI – '0' indicates value below limit, '1' above
%I+10	Input Ch #2 ALARM LO – '0' indicates value above limit, '1' below
%I+11	Input Ch #2 ALARM HI – '0' indicates value below limit, '1' above
%I+12	Input Ch #3 ALARM LO – '0' indicates value above limit, '1' below
%I+13	Input Ch #3 ALARM HI – '0' indicates value below limit, '1' above
%I+14	Input Ch #4 ALARM LO – '0' indicates value above limit, '1' below
%I+15	Input Ch #4 ALARM HI – '0' indicates value below limit, '1' above

The third eight locations (available for %I SIZE values 24)

%I Locations	Description
%I+16	Output Ch #1 BROKEN WIRE 0 = OK, 1 = Wire Broken (I modes only)
%I+17	Output Ch #2 BROKEN WIRE 0 = OK, 1 = Wire Broken (I modes only)
%I+18 .. %I+23	Reserved for future modules. Not used in this module

One of four input or output ranges can be selected. Two of which are voltage ranges. The default range is 0–10V, where input or output voltage values range from 0 to 10 volts. In input mode they report 0 to 32000 integer values to the 90-30 CPU and in output mode values between 0 and 32000 are sent to the module. The –10 to +10V range, values between –32000 to 32000 are sent or received from the CPU over an input voltage range of –10 to +10V. The two current ranges are 4 to 20 mA, and 0 to 20 mA. In each of the current ranges, values between 0 and 32000 are reported back from the module to sent to the module for the entire range.

The following tables shows values sent from the CPU to the module for the Output channels:

Range	Module Mode	*Allowed Values	Sent values from CPU
0 to 10 V	Voltage	0 to 32767	0 to 32767
−10 to 10 V	Voltage	− 32768 to 32767	−32768 to 32767
4 to 20 mA	Current	0 to 32000*	0 to 32767
0 to 20 mA	Current	0 to 32767	0 to 32767

*The phrase, *Allowed Values*, refers to the values that are valid. If a user sends a value > 32000, the module will truncate that value to 32000 before sending it to the DAC.

The following table shows values sent from the Module back to the PLC for the Input channels

Range	Module Mode	Sent values to CPU
0 to 10 V	Voltage	0 – 32767
−10 to 10 V	Voltage	−32768 – 32767
4 to 20 mA	Current	0 – 32767
0 to 20 mA	Current	0 – 32767

The ALARM LO and ALARM HI data fields allow the user to enter values that cause 'alarm' indications to be passed to the PLC. Each channel has a low limit alarm value (ALARM LO) and a high limit alarm value (ALARM HI). These alarm values cause %I points to be set as indicated in the tables above. Values can be entered in all high and low limit fields, even though those channels may not be enabled. Values entered without a sign are assumed to be positive. Value checking should be done to determine if the ALARM LO and ALARM HI values are allowable for the appropriate RANGE. The allowable values are:

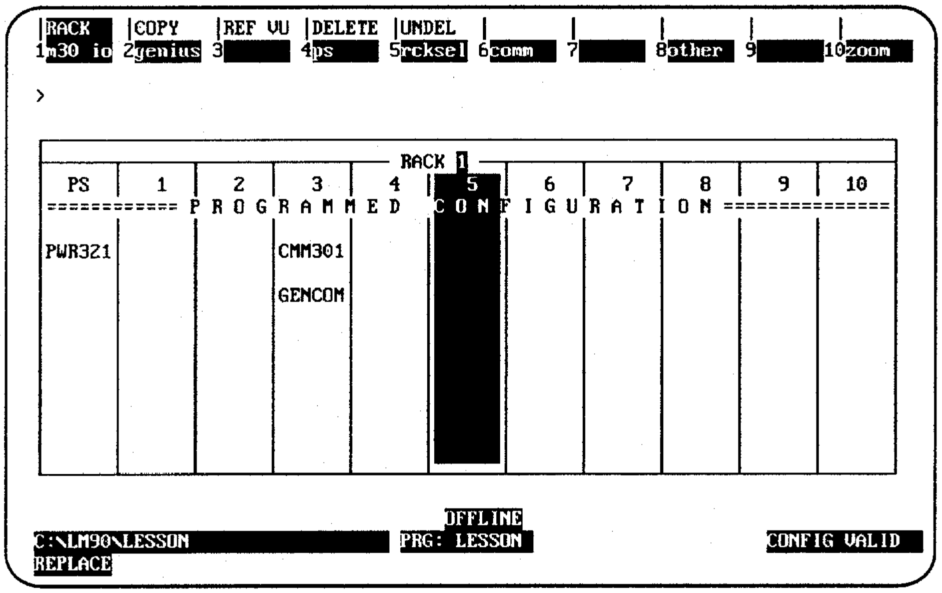
RANGE	Possible limit values
4−20 mA	0..32760
0−20 mA	0..32760
0−10V	0..32760
−10 – +10V	−32768..32760

Section 18: Configuring a Third-Party Module

This section describes configuration of third-party modules. Reference addresses can be configured for %I, %Q, %AI, and %AQ, in addition to a module ID and 16 bytes of hexadecimal data.

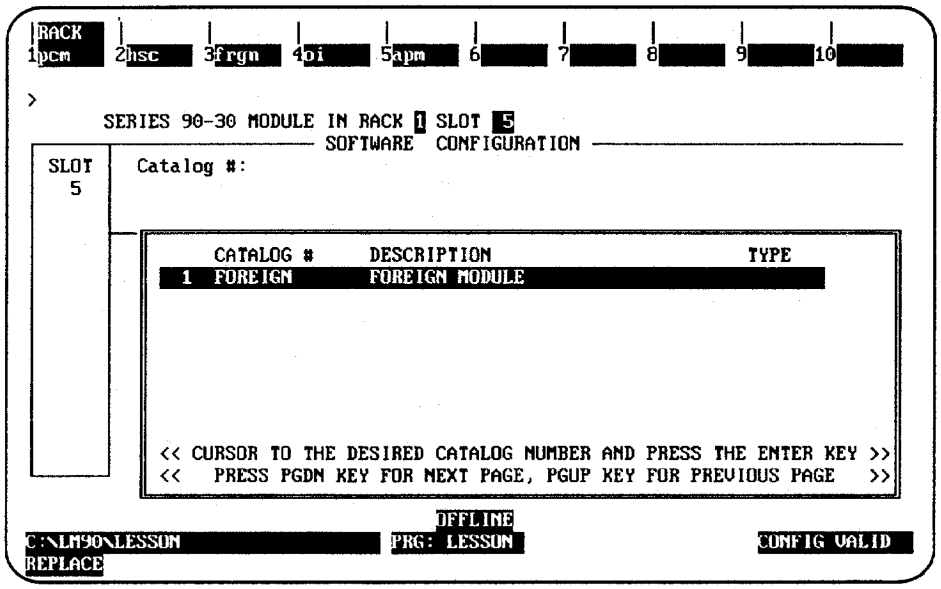
To configure a third-party module on the I/O Configuration Rack screen:

1. Move the cursor to the desired rack and slot location. The slot may be either unconfigured or previously configured.



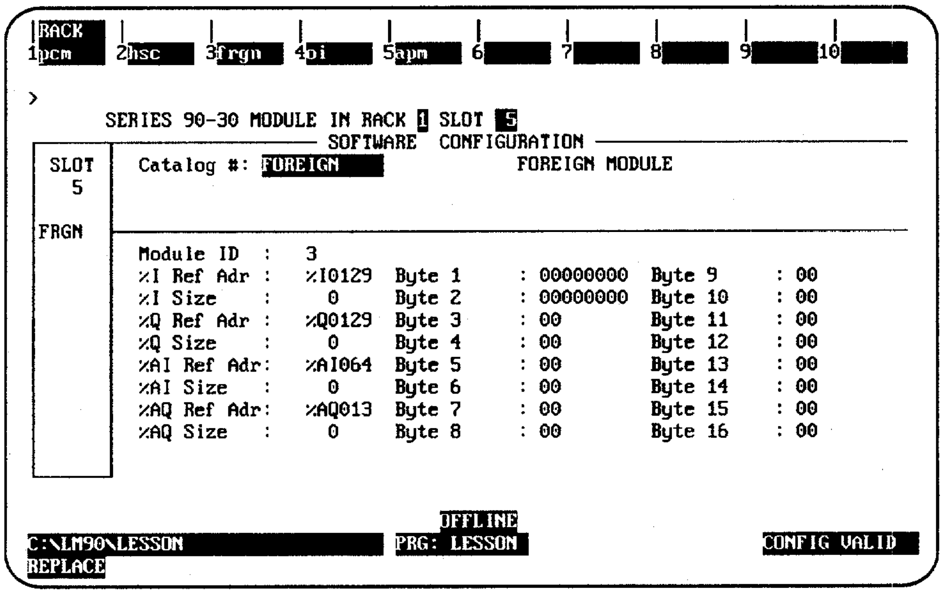
```
|RACK   |COPY   |REF VU |DELETE |UNDEL  |       |       |       |       |
1m30 io 2genius 3       4ps     5rcksel 6comm   7       8other  9       10zoom
>

                                  RACK 1
  PS    |  1  |  2  |  3   |  4  |  5  |  6  |  7  |  8  |  9  |  10
======= P R O G R A M M E D   C O N F I G U R A T I O N ===========
PWR321              CMM301
                    GENCOM

                                  OFFLINE
C:\LM90\LESSON                    PRG: LESSON             CONFIG VALID
REPLACE
```

2. Press **Other** (`F8`) and then **Foreign** (`F3`) from the I/O Configuration Rack screen to display the list of third-party modules.



```
|RACK   |       |       |       |       |       |       |       |       |
1pcm    2hsc    3frgn   4oi     5apm    6       7       8       9       10
>
      SERIES 90-30 MODULE IN RACK 0 SLOT 5
                          SOFTWARE  CONFIGURATION
 SLOT   Catalog #:
  5

           CATALOG #      DESCRIPTION                      TYPE
         1  FOREIGN       FOREIGN MODULE

   << CURSOR TO THE DESIRED CATALOG NUMBER AND PRESS THE ENTER KEY >>
   <<   PRESS PGDN KEY FOR NEXT PAGE, PGUP KEY FOR PREVIOUS PAGE   >>
                                  OFFLINE
C:\LM90\LESSON                    PRG: LESSON             CONFIG VALID
REPLACE
```

3. Press the **Enter** key to display the detail screen.



```
 RACK
1pcm     2hsc     3frgn    4oi      5apm     6        7        8        9        10

>
      SERIES 90-30 MODULE IN RACK 0 SLOT 5
                        SOFTWARE  CONFIGURATION
 SLOT   Catalog #: FOREIGN                 FOREIGN MODULE
  5

FRGN
        Module ID  :    3
        %I Ref Adr :   %I0129  Byte 1    : 00000000  Byte 9     : 00
        %I Size    :     0     Byte 2    : 00000000  Byte 10    : 00
        %Q Ref Adr :   %Q0129  Byte 3    : 00        Byte 11    : 00
        %Q Size    :     0     Byte 4    : 00        Byte 12    : 00
        %AI Ref Adr:   %AI064  Byte 5    : 00        Byte 13    : 00
        %AI Size   :     0     Byte 6    : 00        Byte 14    : 00
        %AQ Ref Adr:   %AQ013  Byte 7    : 00        Byte 15    : 00
        %AQ Size   :     0     Byte 8    : 00        Byte 16    : 00

                                  OFFLINE
C:\LM90\LESSON                    PRG: LESSON                    CONFIG VALID
REPLACE
```

Parameter	Description
Module ID	A 1, 2, or 3-digit signed integer value representing a vendor module designation. A range of valid vendor module IDs is used to allow validation.
Reference Data	Enter the starting address and length for each of the %I, %Q, %AI, and %AQ references. The reference address parameters default to the next highest reference address. The size parameters default to zero and are validated to be within the configuration memory limits and the configured CPU for the respective reference type.
Soft Switch Data	Sixteen bytes of soft switch data can be configured. Bytes 1 and 2 are binary values (00000000 to 11111111). Bytes 3 through 16 are byte hexadecimal values (00 to FF).

Section 19: Configuration Reference View

The configuration reference view feature enables you to view tabular displays of configured modules with the same reference (%I, %Q, %AI, %AQ, %G, or %R). This feature can be helpful when assigning new reference addresses or resolving address conflicts, such as overlapping.

Note

The configuration reference view feature is not available for the CPU 211.

The reference view table is sorted in ascending order by user reference, with the lowest address listed first. Mixed discrete inputs and outputs (%QI) are shown in both the %Q and %I tables. Mixed analog inputs and outputs (%AQI) are shown in both the %AQ and %AI tables.

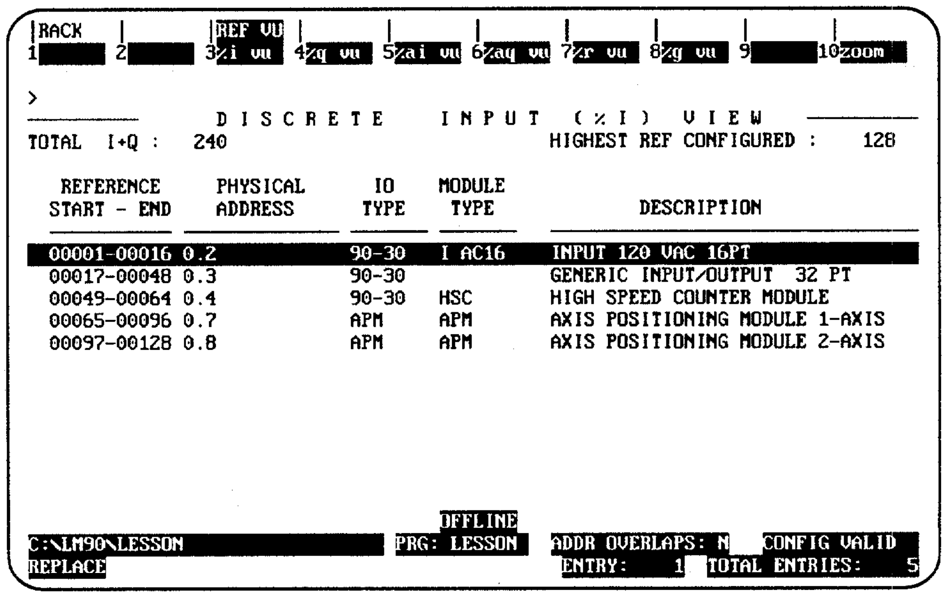
Data contained in the reference view table may only be viewed; it cannot be edited. Editing may only be done on the detail screens. Modules configured from the rack screens have a reference address assigned to them. When a module is configured, it is automatically entered into the Reference View table for its reference type. When a module is deleted, it is automatically removed from the table.

When the reference address of a module is changed, the reference view table is updated automatically.

Displaying the Reference View Table

The Reference View screen for the current module reference type may be displayed by pressing **Reference View** (**`Shift-F3`**) from any rack or detail screen in the I/O configuration rack function. From the detail screen, the cursor will be positioned at that same module on the Reference View screen.

To display a different Reference View screen, after pressing **Shift-F3** to select the reference view function, press the appropriate function key (F3 through F8) for the view screen you wish to display. For example, to display the %I Discrete Input Reference screen shown below, press **Reference View** (**`Shift-F3`**) from an input module detail screen, or **Shift-F3** and then **%I View** (**`F3`**) from the detail screen of another reference type.


```
|RACK     |         |REF VU  |        |        |        |        |        |        |
1         2         3%i vu   4%q vu   5%ai vu  6%aq vu  7%r vu   8%g vu   9        10zoom

>
                DISCRETE     INPUT  (%I)  VIEW
TOTAL  I+Q :   240                          HIGHEST REF CONFIGURED :    128

  REFERENCE     PHYSICAL      IO     MODULE
 START - END    ADDRESS       TYPE   TYPE              DESCRIPTION

 00001-00016 0.2              90-30  I AC16     INPUT 120 VAC 16PT
 00017-00048 0.3              90-30             GENERIC INPUT/OUTPUT  32 PT
 00049-00064 0.4              90-30  HSC        HIGH SPEED COUNTER MODULE
 00065-00096 0.7              APM    APM        AXIS POSITIONING MODULE 1-AXIS
 00097-00128 0.8              APM    APM        AXIS POSITIONING MODULE 2-AXIS



                                     OFFLINE
C:\LM90\LESSON                      PRG: LESSON  ADDR OVERLAPS: N   CONFIG VALID
REPLACE                                          ENTRY:     1  TOTAL ENTRIES:   5
```

The first line of the display area clearly identifies the reference type of this view. On the second line, ***Total Used*** on an analog reference view screen shows the total number of references configured. On a discrete reference view screen, ***Total I+Q*** shows the combined total. The highest reference number configured is also displayed on the second line.

Each entry in the reference view table contains the following information:

Field	Description
Reference Start – End	The starting and ending reference addresses for the module.
Physical Address	The PLC hardware physical address for the module. Each entry in this column will contain the rack and slot information, each separated by a period (for example, rack.slot).
I/O Type	The input/output type which controls this module.
Module Type	The type and size of the module.
Description	The description field.

If overlaps exist in the table, a **Y** (Yes) is displayed on the second status line at the bottom of the screen. If there are no overlaps, an **N** (No) is displayed.

The position of the selected entry is displayed in the *Entry* field, and the number of reference table entries is displayed in the *Total Entries* field. For example, in the previous screen, the first entry is the selected entry, and the total number of entries is five.

Moving the Cursor

Use the Up and Down cursor movement keys to scroll between rows of the Reference View table. If the cursor is at the beginning or end of the table, pressing these keys will scroll additional information not already displayed, one single row at a time. The cursor field will always include an entire module (e.g., a single row).

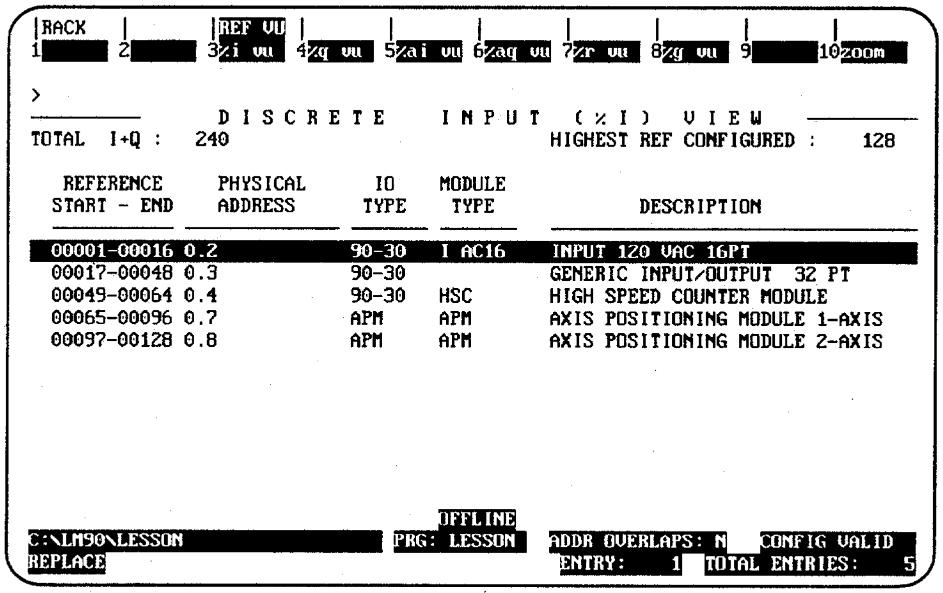
Press the **Home** key to position the cursor at the beginning of the table, or the **End** key to go to the end of the table.

If all the configured modules in the PLC cannot be displayed on a single screen, press the **Page Down** key to view additional entries, or the **Page Up** key to view previous entries.

Displaying the Detail Screen

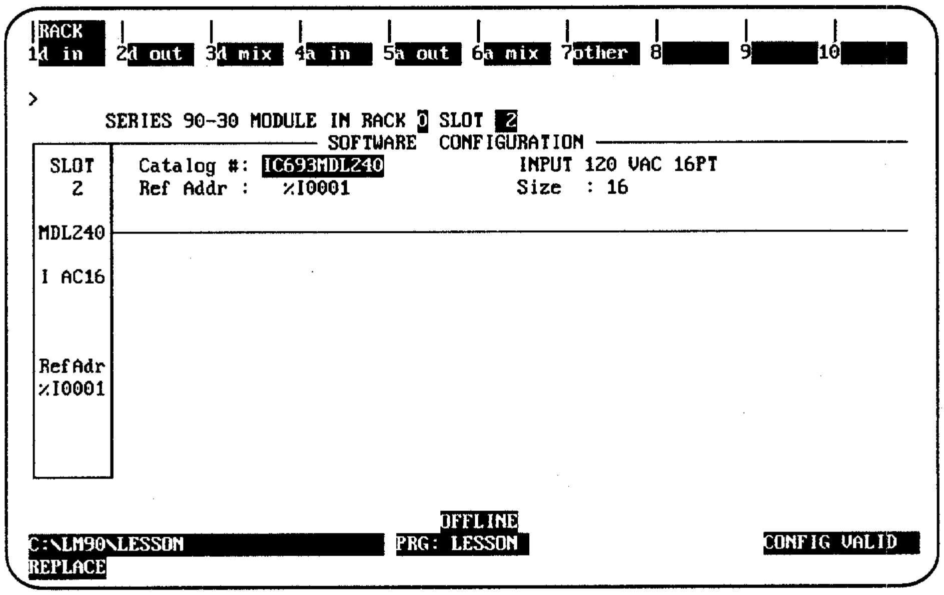
All of the Reference View screens have a **Zoom (F10)** function key displayed at the top of the screen. This key may be used to go directly to the detailed module screen for the module highlighted on the Reference View screen.

In the following example, the 16-point input module with a starting reference address at 00001 is shown in reverse video.



```
|RACK    |        |REF VU |        |        |        |        |        |        |
1        2        3%i vu  4%q vu   5%ai vu  6%aq vu  7%r vu   8%g vu   9        10zoom

>
__________     D I S C R E T E    I N P U T   ( % I )   V I E W  __________
TOTAL  I+Q :   240                              HIGHEST REF CONFIGURED :    128

  REFERENCE     PHYSICAL         IO     MODULE
 START - END    ADDRESS          TYPE   TYPE               DESCRIPTION
               _____________    ______  ______    _________________________________
 00001-00016 0.2                 90-30  I AC16    INPUT 120 VAC 16PT
 00017-00048 0.3                 90-30            GENERIC INPUT/OUTPUT  32 PT
 00049-00064 0.4                 90-30  HSC       HIGH SPEED COUNTER MODULE
 00065-00096 0.7                 APM    APM       AXIS POSITIONING MODULE 1-AXIS
 00097-00128 0.8                 APM    APM       AXIS POSITIONING MODULE 2-AXIS




                                        OFFLINE
C:\LM90\LESSON                       PRG: LESSON  ADDR OVERLAPS: N   CONFIG VALID
REPLACE                                            ENTRY:       1 TOTAL ENTRIES:    5
```

When the **Zoom (F10)** key is pressed, the detail screen for this module is displayed.



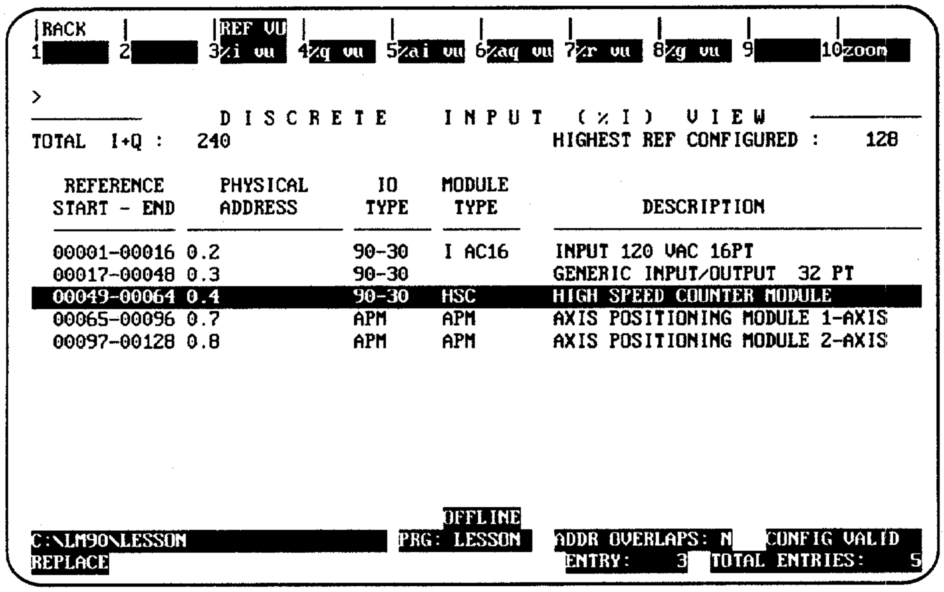
```
|RACK  |       |       |       |       |       |       |       |       |
1d in  2d out  3d mix  4a in   5a out  6a mix  7other  8       9       10

>
      SERIES 90-30 MODULE IN RACK 0 SLOT 2
                        SOFTWARE  CONFIGURATION
 SLOT | Catalog #: IC693MDL240          INPUT 120 VAC 16PT
  2   | Ref Addr :   %I0001             Size  : 16
      |
MDL240|
      |
I AC16|
      |
      |
RefAdr|
%I0001|

                                   OFFLINE
C:\LM90\LESSON                     PRG: LESSON                CONFIG VALID
REPLACE
```

To return to the Reference View screen, press **Reference View (Shift-F3)**.

Displaying the Rack Screen

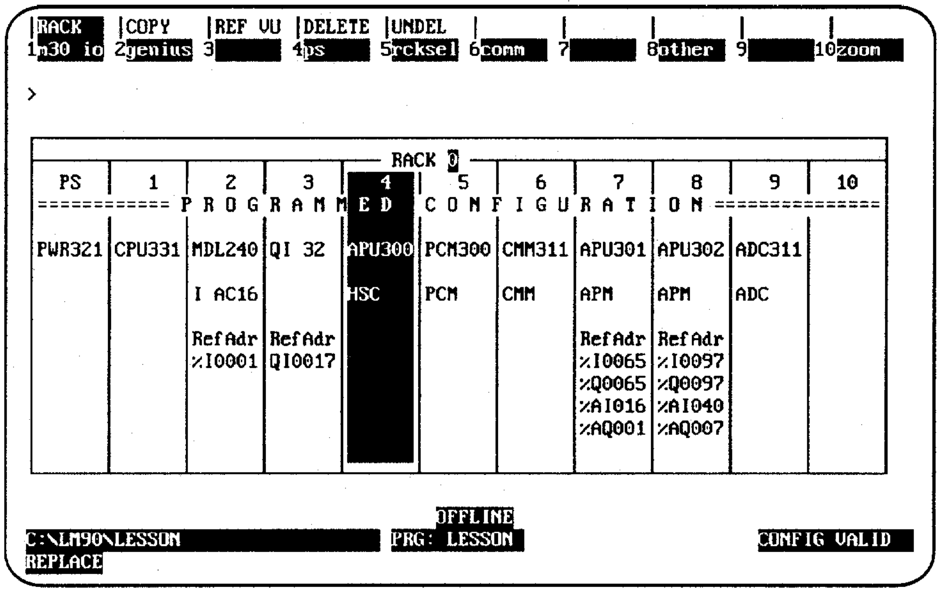
All of the Reference View screens have a **Rack (Shift-F1)** function key displayed at the top of the screen. This key may be used to go directly to the Rack screen from the reference view function.

In the following example, the High Speed Counter module is shown in reverse video. This module is physically located in slot 4 of the main rack.



```
|RACK  |      |REF VU |       |       |       |       |       |       |
1      2      3%i vu  4%q vu  5%ai vu 6%aq vu 7%r vu  8%g vu  9       10zoom

>
             D I S C R E T E     I N P U T  ( % I )  V I E W
TOTAL  I+Q :  240                          HIGHEST REF CONFIGURED :    128

  REFERENCE    PHYSICAL       IO    MODULE
  START - END  ADDRESS        TYPE   TYPE            DESCRIPTION

  00001-00016 0.2             90-30  I AC16    INPUT 120 VAC 16PT
  00017-00048 0.3             90-30            GENERIC INPUT/OUTPUT  32 PT
  00049-00064 0.4             90-30  HSC       HIGH SPEED COUNTER MODULE
  00065-00096 0.7             APM    APM       AXIS POSITIONING MODULE 1-AXIS
  00097-00128 0.8             APM    APM       AXIS POSITIONING MODULE 2-AXIS

                                    OFFLINE
C:\LM90\LESSON                      PRG: LESSON  ADDR OVERLAPS: N  CONFIG VALID
REPLACE                                           ENTRY:     3  TOTAL ENTRIES:   5
```

When **Rack** **(Shift-F1)** is pressed from the Reference View screen, the screen will display the main rack and highlight the slot of the module selected on the Reference View screen (in this example, slot 4).

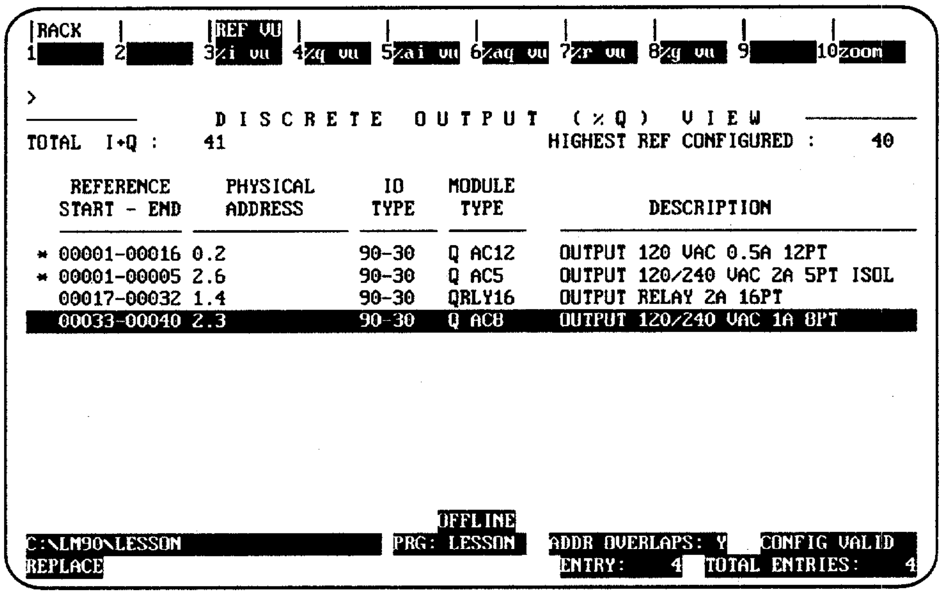


```
|RACK    |COPY    |REF VU  |DELETE  |UNDEL   |        |        |        |        |
1m30 io  2genius  3        4ps      5rcksel  6comm    7        8other   9        10zoom

>

                                   RACK 0
  PS    |   1    |   2    |   3    |   4    |   5    |   6    |   7    |   8    |   9    |   10
=========== P R O G R A M M E D   C O N F I G U R A T I O N ==============

PWR321  |CPU331  |MDL240  |QI 32   |APU300  |PCM300  |CMM311  |APU301  |APU302  |ADC311  |
                  I AC16            HSC      PCM      CMM      APM      APM      ADC
                  RefAdr   RefAdr                              RefAdr   RefAdr
                  %I0001   QI0017                              %I0065   %I0097
                                                               %Q0065   %Q0097
                                                               %AI016   %AI040
                                                               %AQ001   %AQ007

                                        OFFLINE
C:\LM90\LESSON                        PRG: LESSON                       CONFIG VALID
REPLACE
```

To return to the Reference View screen, press **Reference View** **(Shift-F3)**.

Overlapping References

Overlapping references will result in an invalid configuration. When the reference address assigned to a module overlaps with another module's reference address of the same reference type, an asterisk (*) is displayed at the beginning of the row for both modules. If the overlap is fatal, resulting in CONFIG INVALID, double asterisks (**) are displayed.



```
|RACK   |      |REF VU |      |      |      |      |      |      |
1       2      3%i vu  4%q vu 5%ai vu 6%aq vu 7%r vu 8%g vu 9      10zoom

>
 ________      D I S C R E T E   O U T P U T   ( % Q )   V I E W   ________
TOTAL  I+Q :   41                          HIGHEST REF CONFIGURED :      40

   REFERENCE     PHYSICAL        IO     MODULE
   START - END   ADDRESS         TYPE    TYPE              DESCRIPTION

 * 00001-00016 0.2               90-30   Q AC12    OUTPUT 120 VAC 0.5A 12PT
 * 00001-00005 2.6               90-30   Q AC5     OUTPUT 120/240 VAC 2A 5PT ISOL
   00017-00032 1.4               90-30   QRLY16    OUTPUT RELAY 2A 16PT
   00033-00040 2.3               90-30   Q AC8     OUTPUT 120/240 VAC 1A 8PT



                                       OFFLINE
C:\LM90\LESSON                      PRG: LESSON  ADDR OVERLAPS: Y  CONFIG VALID
REPLACE                                           ENTRY:     4  TOTAL ENTRIES:   4
```

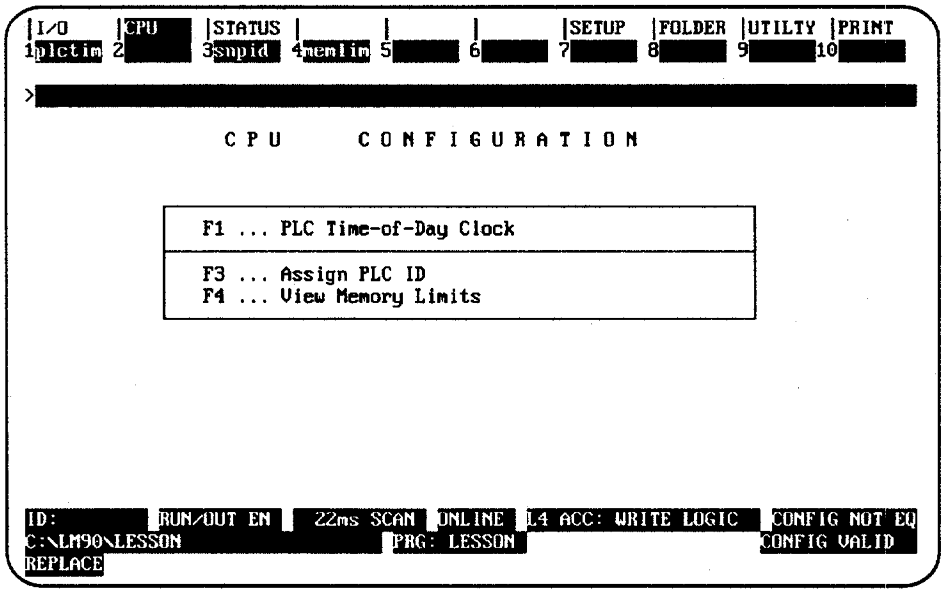
To resolve the conflicting overlap, position the cursor on the reference address which must be changed and press **Zoom** (**F10**). The detail module screen will be displayed, allowing the address to be adjusted.

Caution

When adjusting reference addresses, be careful not to cause another overlap.

Chapter 11 CPU Configuration

Use the CPU configuration function to set the operating characteristics of the CPU. To access the CPU configuration functions, press **CPU (F2)** from the main menu. Then continue below.



```
|I/O     |CPU     |STATUS |        |        |        |SETUP   |FOLDER  |UTILTY  |PRINT
1plctim 2        3snpid  4memlim 5        6        7        8        9        10
>

                 C P U       C O N F I G U R A T I O N

              F1 ... PLC Time-of-Day Clock

              F3 ... Assign PLC ID
              F4 ... View Memory Limits

ID:            RUN/OUT EN    22ms SCAN  ONLINE  L4 ACC: WRITE LOGIC    CONFIG NOT EQ
C:\LM90\LESSON                          PRG: LESSON                    CONFIG VALID
REPLACE
```

Function Key	Function	Description	Page
F1	PLC Time	View the computer's time-of-day clock, or view and change the PLC time-of-day clock.	11-2
F3	SNP ID	Assign a new value to the Series Ninety Protocol (SNP) ID.	11-4
F4	Memory Limits	Display the CPU memory allocation.	11-5

These functions are described on the pages that follow.

Storing the CPU Configuration to the PLC

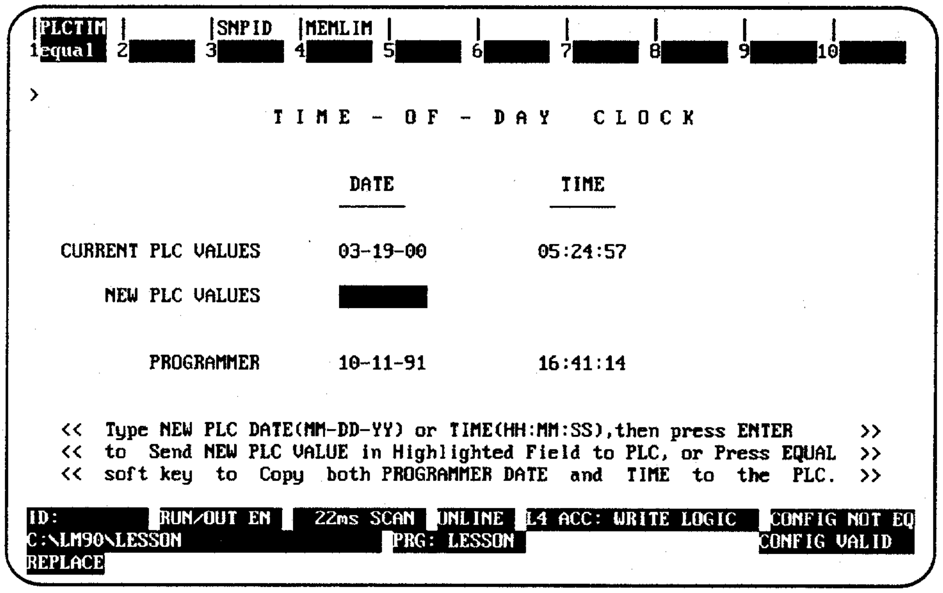
Use the program utility functions (in **ONLINE** mode) to store configuration data to the PLC. For instructions, refer to chapter 8, "Program Utilities."

PLC Date and Time

To display the current date and time, press **PLC Time** `(F1)`.

Note

The PLC time-of-day clock function is only available with the Model 331 and higher CPUs. It is not available with the Model 311 or Model 313 CPU. If the **PLCTIM** `(Shift-F1)` key is pressed, the following screen is disabled and the software displays a message indicating that the function is not available.



```
|PLCTIM |       |SNPID  |MEMLIM |       |       |       |       |       |
1equal  2       3       4       5       6       7       8       9       10

>
                    T I M E  -  O F  -  D A Y    C L O C K

                                    DATE               TIME
                                   ______             ______

   CURRENT PLC VALUES              03-19-00           05:24:57

       NEW PLC VALUES

           PROGRAMMER              10-11-91           16:41:14

  <<  Type NEW PLC DATE(MM-DD-YY) or TIME(HH:MM:SS),then press ENTER     >>
  <<  to  Send NEW PLC VALUE in Highlighted Field to PLC, or Press EQUAL >>
  <<  soft key  to  Copy  both PROGRAMMER DATE  and  TIME  to  the  PLC. >>

ID:          RUN/OUT EN   22ms SCAN  ONLINE  L4 ACC: WRITE LOGIC   CONFIG NOT EQ
C:\LM90\LESSON                      PRG: LESSON                    CONFIG VALID
REPLACE
```

Note

When this screen is first displayed, the time-of-day clock is read from the PLC. Elapsed time is tracked by the Logicmaster 90-30/20/Micro software. Other devices, such as CIMPLICITY 90-ADS, may change the time-of-day clock, but the change will not be reflected on this screen until you exit and return to this screen.

The current date and time for the programmer are displayed in all operating modes. To also display the PLC date and time, place the computer in **ONLINE** or **MONITOR** mode. If the programmer is in **OFFLINE** mode or is not communicating with the CPU, asterisks are displayed in the *Current PLC Values* fields.

Changing the PLC Date and Time

To change the date and/or time shown for the PLC, the computer must be in **ONLINE** mode and communicating with the PLC.

To make the PLC date and time the same as the values shown for the computer, press **Equal (F1)**. To use a different date and/or time:

1. Enter the new date, using dash (minus) characters between fields (MM-DD-YY).

Abbreviation	Description
MM	Month from 1 to 12.
DD	Day from 01 to 31.
YY	Year from 00 to 99.

2. Enter the new time, using colons between fields (HH:MM:SS).

Abbreviation	Description
HH	Hours from 00 to 23.
MM	Minutes from 00 to 59.
SS	Seconds from 00 to 59.

Note

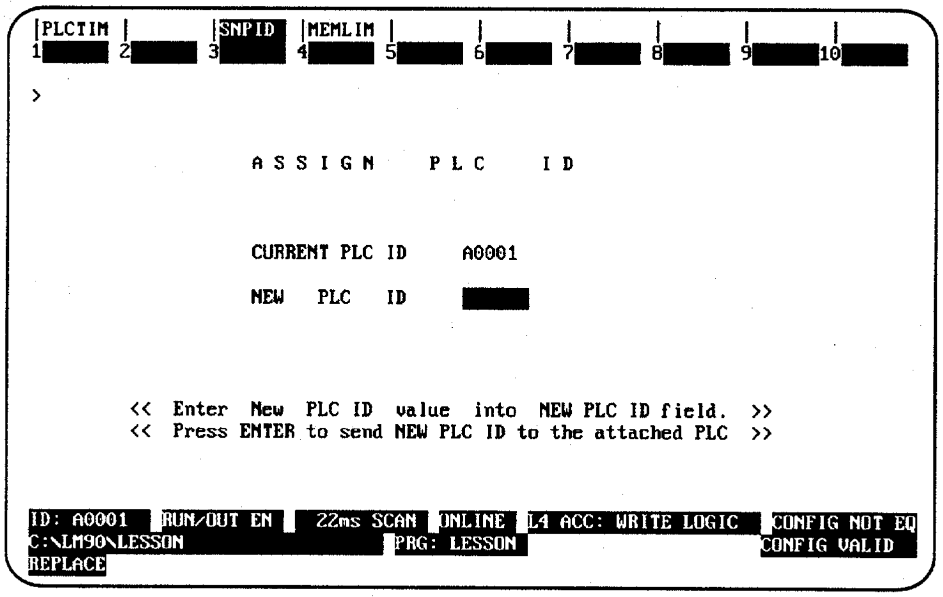
If hours are specified, entering minutes and seconds is optional. Partial times must, however, end with a colon. For example, HH: or HH:MM:

3. Press the **Enter** or **Escape** key. The change in the currently highlighted field is sent to the PLC and will appear in the ***Current PLC Values*** fields on the screen.

SNP ID

For multidrop configurations, each CPU connected to the system must have a unique identification name consisting of 1 to 6 characters. The SNP ID is not required for peer-to-peer communications.

The current SNP ID name is displayed in **ONLINE** or **MONITOR** mode. To display the current SNP ID or assign a new name, press SNP ID **(F3)**. The entry for the *New SNP ID* field can only be changed in **ONLINE** mode with communications. If the programmer is in **OFFLINE** mode or is not communicating with the CPU, asterisks are displayed in the *Current SNP ID* field.


```
|PLCTIM |      |SNPID |MEMLIM |      |      |      |      |      |
1        2      3      4      5      6      7      8      9      10
>

                 A S S I G N    P L C    I D

                 CURRENT PLC ID     A0001

                 NEW   PLC   ID

        << Enter New PLC ID value into NEW PLC ID field. >>
        << Press ENTER to send NEW PLC ID to the attached PLC >>

ID: A0001   RUN/OUT EN   22ms SCAN  ONLINE L4 ACC: WRITE LOGIC   CONFIG NOT EQ
C:\LM90\LESSON                      PRG: LESSON                  CONFIG VALID
REPLACE
```

Changing the SNP ID Name

To change the identification of the PLC:

1. Enter a new name in the *New SNP ID* field, using the alphanumeric characters (A – Z, 0 – 9) or special characters (–, @, _, #, $, %, <, >, =, +, &). The first character must be alphabetic.

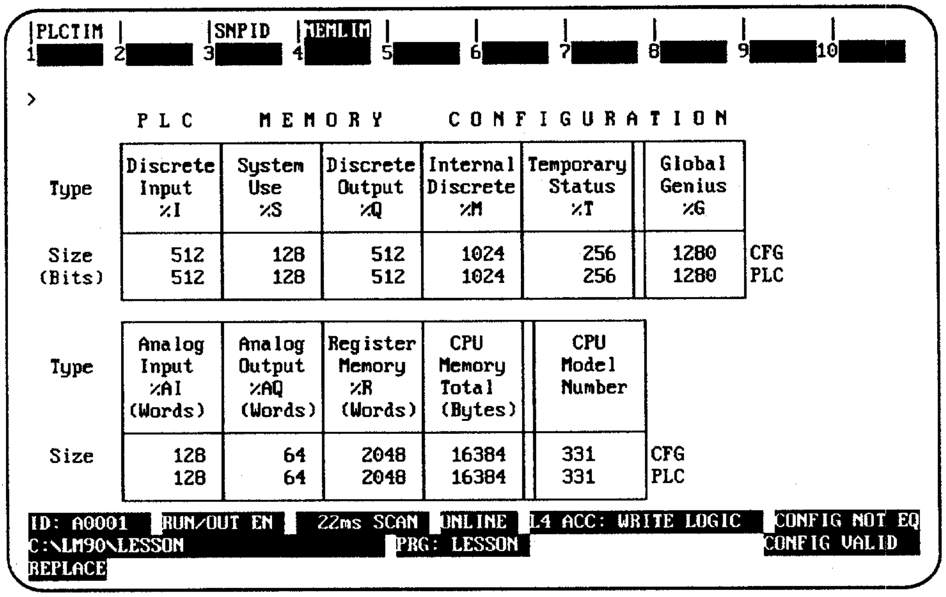
Note

Lower-case characters may be entered using the Hand-Held Programmer; however, Logicmaster 90-30/20/Micro software supports only upper-case characters.

2. To clear the SNP ID, enter all blank characters.
3. Press the **Enter** key to send this new name to the attached SNP device.

PLC Memory Limits

To display the reference and logic memory limits for the PLC, press **Memory Limits** **(F4)**. The limits are fixed, based on the model of the CPU. This screen shows the fixed memory limits for a CPU 331.



```
|PLCTIM |      |SNPID  |MEMLIM |      |      |      |      |      |
1       2      3       4       5      6      7      8      9      10
>
        P L C      M E M O R Y      C O N F I G U R A T I O N
```

	Discrete Input %I	System Use %S	Discrete Output %Q	Internal Discrete %M	Temporary Status %T	Global Genius %G	
Size (Bits)	512	128	512	1024	256	1280	CFG
	512	128	512	1024	256	1280	PLC

	Analog Input %AI (Words)	Analog Output %AQ (Words)	Register Memory %R (Words)	CPU Memory Total (Bytes)	CPU Model Number	
Size	128	64	2048	16384	331	CFG
	128	64	2048	16384	331	PLC

```
ID: A0001  RUN/OUT EN   22ms SCAN  ONLINE  L4 ACC: WRITE LOGIC  CONFIG NOT EQ
C:\LM90\LESSON                     PRG: LESSON                  CONFIG VALID
REPLACE
```

The current memory allocations stored in the program folder are displayed in all operating modes. To display the PLC values, place the computer in **ONLINE** or **MONITOR** mode.

The upper portion of this screen shows the maximum values for discrete references (%I, %Q, %M, %T, %S, and %G). These values are set by the system and will change only when the CPU model number is changed.

The lower portion of this screen shows values for register references (%AI, %AQ, and %R). These values are also set by the system and will change only when the CPU model number is changed. The sum of CPU memory, according to the particular CPU model is displayed in the *CPU Memory Total* field.

Appendix A Programming Lesson

This appendix contains a tutorial for the programming software package. The lesson includes:

- Creating a program folder.
- Creating a program.
- Entering a variable declaration.
- Adding ladder logic to the program.
- Printing the program.
- Exiting the programmer.

Help Screens

Logicmaster 90-30/20/Micro software includes detailed Help screens. These Help screens are loaded onto the hard disk of your programmer during the software installation procedure and are readily accessible. To access the Help screens, press **ALT-H** for help, **ALT-I** for instruction mnemonic help, or **ALT-K** for key help.

Starting the Lesson

Before you can start, the programming software must be installed and started. If that has not been done yet, please turn back to chapter 2, "Operation," for instructions.

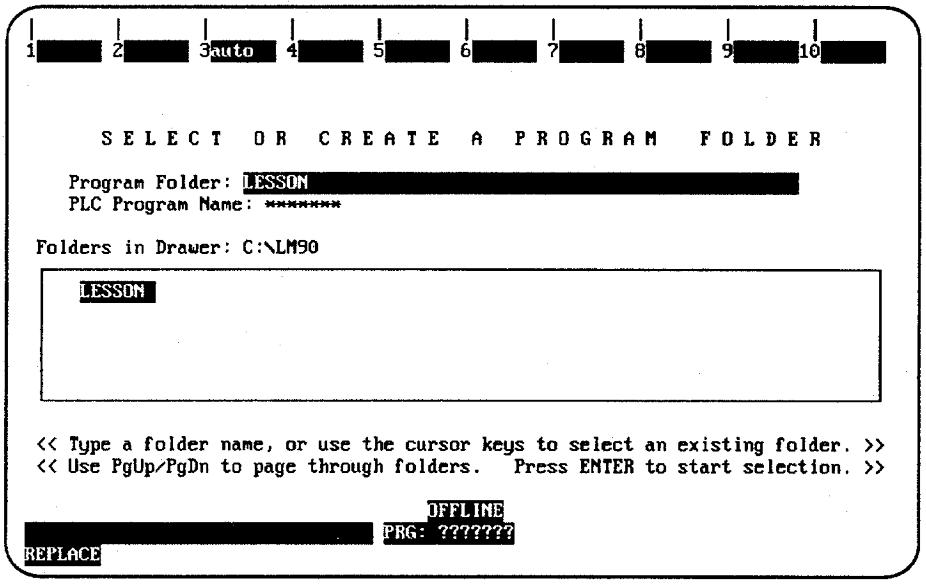
Exiting the Programmer

You can exit the programmer at any time by pressing the **CTRL-Break** keys.

Creating a Program Folder

For this lesson, you will create and use a program folder named **LESSON**.

When Logicmaster 90-30/20/Micro software is started (if the current default directory is an existing program folder), a screen similar to the one shown below appears. If the software is already running, you can display a similar screen from the Program Folder menu by pressing **Folder** (**F8** or **Shift-F8**) from the Programming Software main menu and then **Select** (**F1**).

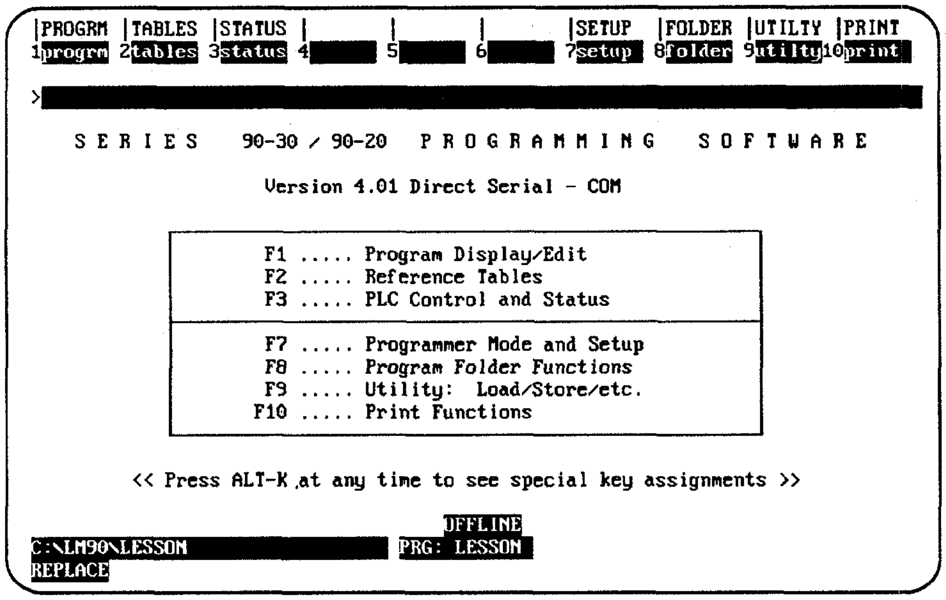


```
SELECT OR CREATE A PROGRAM FOLDER

  Program Folder: LESSON
  PLC Program Name: ********

Folders in Drawer: C:\LM90

    LESSON

<< Type a folder name, or use the cursor keys to select an existing folder. >>
<< Use PgUp/PgDn to page through folders.   Press ENTER to start selection. >>

                                        OFFLINE
                                        PRG: ???????
REPLACE
```

1. If the current default directory is not an existing program folder, type in a name of seven characters or less for the folder. For this lesson, type **LESSON**.
2. Press the **Enter** key. The following prompt will appear at the top of the screen:

```
Program folder does not exist; create new folder?  (Y/N)
```

3. To create the program folder, enter **Y** (Yes). The software will accept the name and display the Programming Software main menu.

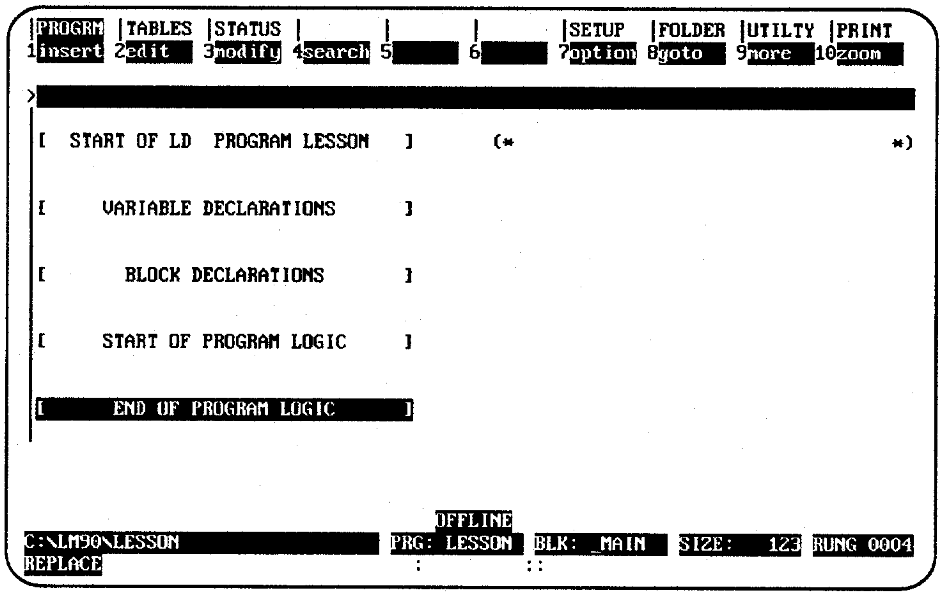


```
|PROGRM |TABLES |STATUS |        |        |        |SETUP  |FOLDER |UTILTY |PRINT
1progrm 2tables 3status 4        5        6        7setup  8folder 9utilty10print
>
   S E R I E S    90-30 / 90-20   P R O G R A M M I N G    S O F T W A R E

                    Version 4.01 Direct Serial - COM

              F1 ..... Program Display/Edit
              F2 ..... Reference Tables
              F3 ..... PLC Control and Status

              F7 ..... Programmer Mode and Setup
              F8 ..... Program Folder Functions
              F9 ..... Utility:  Load/Store/etc.
             F10 ..... Print Functions

       << Press ALT-K at any time to see special key assignments >>

                                          OFFLINE
C:\LM90\LESSON                           PRG: LESSON
REPLACE
```

The name of your new program folder (**LESSON**) should be displayed at the bottom of the screen.

Creating a Program

Create a program in this program folder by pressing **Program (F1)**. The following screen will be displayed:



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3modify 4search 5       6       7option 8goto   9more   10zoom
>
[  START OF LD  PROGRAM LESSON   ]      (*                                    *)

[     VARIABLE DECLARATIONS      ]

[       BLOCK DECLARATIONS       ]

[     START OF PROGRAM LOGIC     ]

[      END OF PROGRAM LOGIC      ]

                                          OFFLINE
C:\LM90\LESSON                         PRG: LESSON  BLK: _MAIN   SIZE:    123  RUNG 0004
REPLACE
```

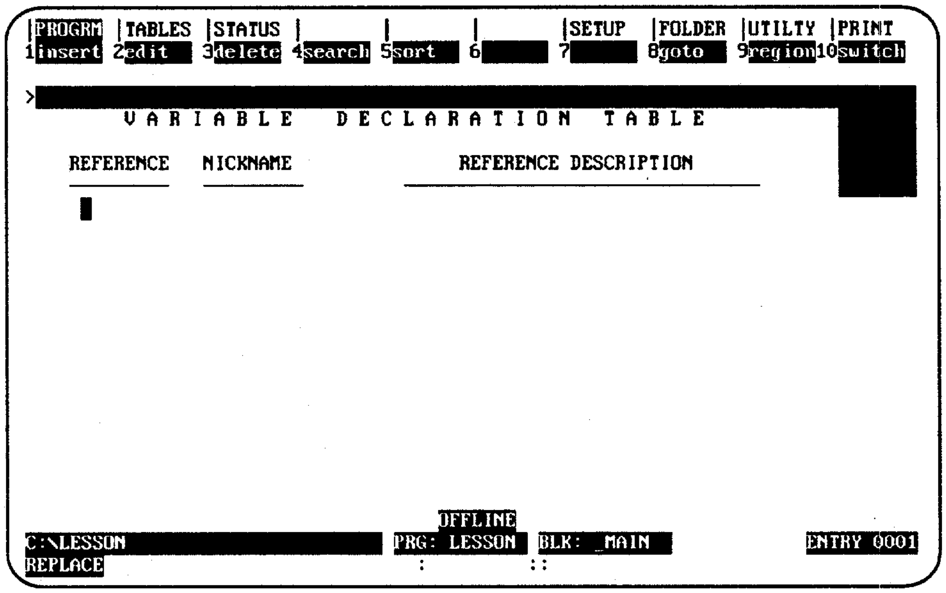
The screen shows a **[VARIABLE DECLARATIONS]** marker where variable declarations can be inserted. Variable declarations are used to list nicknames and reference descriptions for the block. At the _MAIN block, the nicknames listed in the variable declaration table are known throughout all blocks; in a subroutine block, the nicknames are local and known only to that particular block.

Some programs will contain no declarations. This lesson shows how to enter a variable declaration.

Entering a Variable Declaration

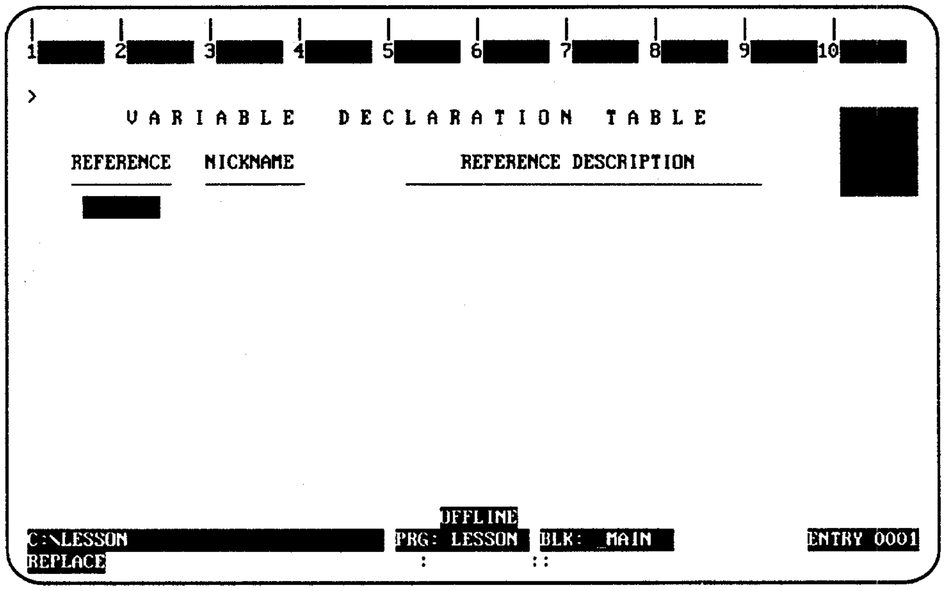
To enter a variable declaration:

1. Use the cursor keys to move the cursor block to the **[VARIABLE DECLARATIONS]** marker.
2. Press **Zoom (F10)** to display a table where variables for the program can be entered.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3delete 4search 5sort   6       7       8goto   9region10switch
>
      V A R I A B L E   D E C L A R A T I O N   T A B L E

   REFERENCE   NICKNAME            REFERENCE DESCRIPTION

                                          OFFLINE
C:\LESSON                               PRG: LESSON  BLK: _MAIN        ENTRY 0001
REPLACE                                   :          ::
```

3. Press **Insert (F1)** to begin entering information in the *Reference* field.

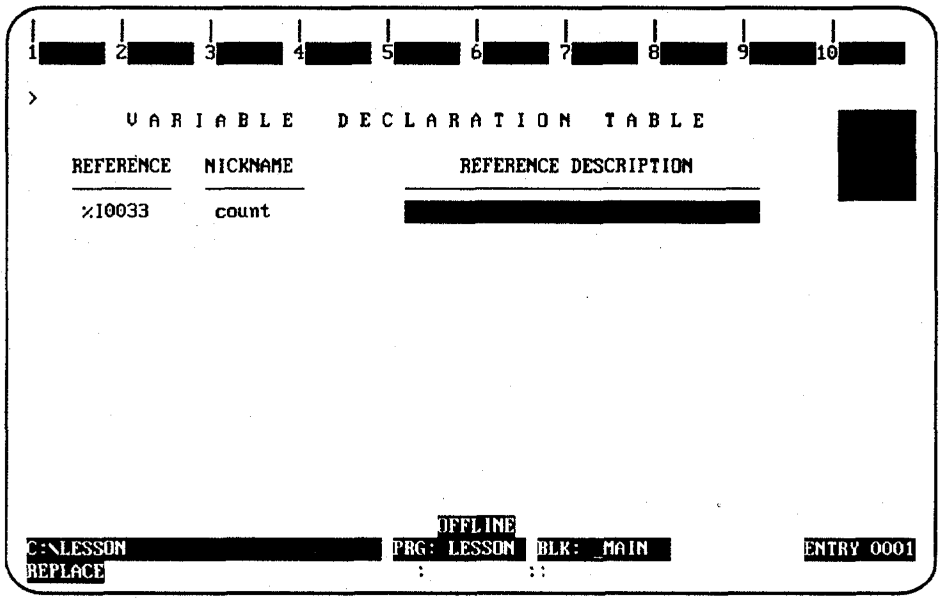


```
|       |       |       |       |       |       |       |       |       |
1       2       3       4       5       6       7       8       9      10
>
      V A R I A B L E   D E C L A R A T I O N   T A B L E

   REFERENCE   NICKNAME            REFERENCE DESCRIPTION

                                          OFFLINE
C:\LESSON                               PRG: LESSON  BLK: _MAIN        ENTRY 0001
REPLACE                                   :          ::
```

4. Enter the machine reference which the nickname will represent. For this lesson, type **`%I33`** or **`33I`**.

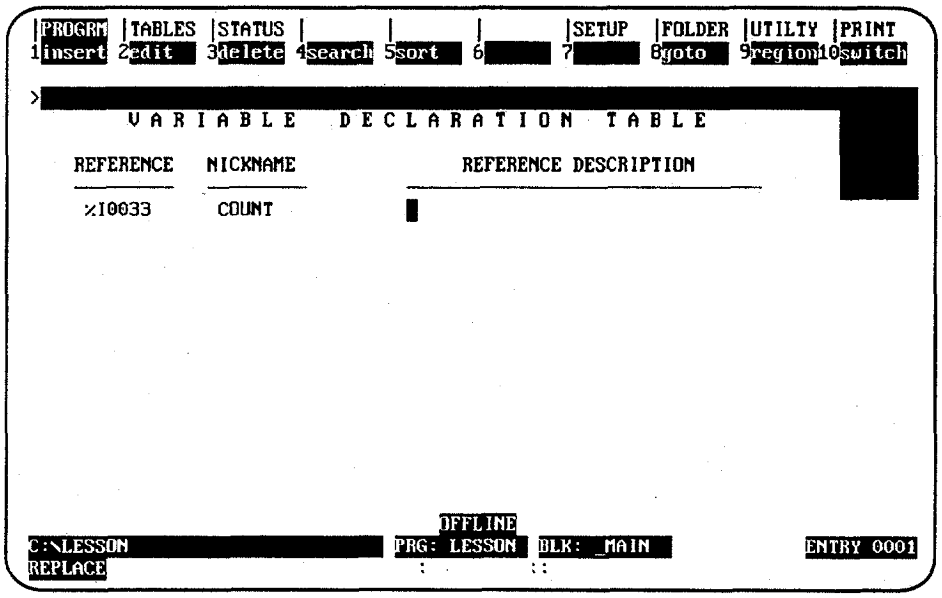
5. Press the **Enter** key. The cursor block moves to the *Nickname* field.

6. Enter the nickname ***count***, and press the **Enter** key. The cursor block moves to the *Reference Description* field.



```
1      2      3      4      5      6      7      8      9      10
>
        V A R I A B L E   D E C L A R A T I O N   T A B L E

     REFERENCE   NICKNAME          REFERENCE DESCRIPTION

      %I0033      count

                                  OFFLINE
C:\LESSON                       PRG: LESSON  BLK: _MAIN         ENTRY 0001
REPLACE                              :          ::
```

7. For this lesson, we will not use the *Reference Description* field. Press the **Escape** key to save your entries and leave **`INSERT`** mode. Note that when you press the **Escape** key, the software automatically changes your entry in the *Nickname* field to uppercase letters.



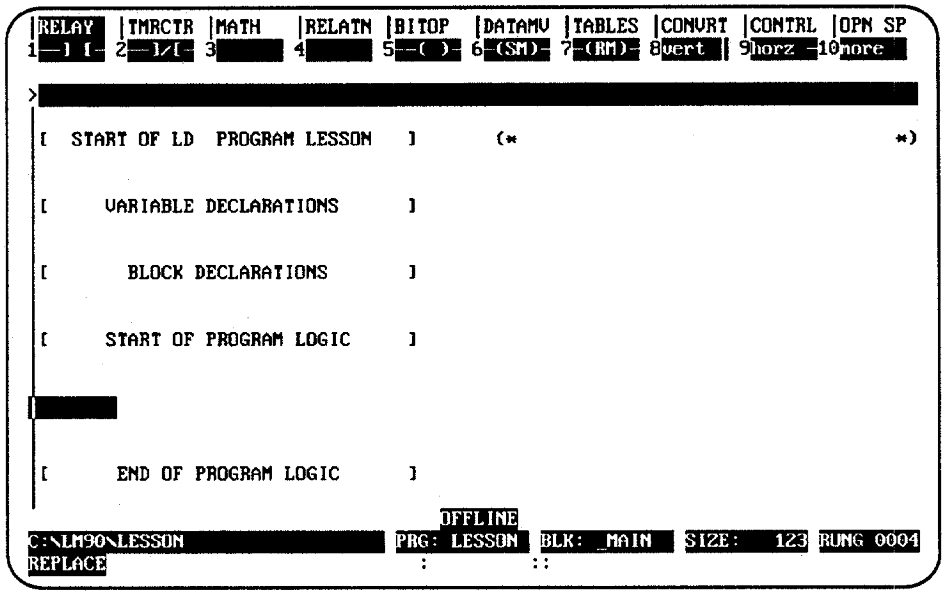
```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3delete 4search 5sort   6       7       8goto   9region10switch
>
        V A R I A B L E   D E C L A R A T I O N   T A B L E

     REFERENCE   NICKNAME          REFERENCE DESCRIPTION

      %I0033      COUNT

                                  OFFLINE
C:\LESSON                       PRG: LESSON  BLK: _MAIN         ENTRY 0001
REPLACE                              :          ::
```

8. Press the **Escape** key again to return to the program ladder logic display.

Adding Ladder Logic to the Program

To create logic for the example program:

1. On the program screen, use the Down cursor key to move the cursor to the **`[END OF PROGRAM LOGIC]`** marker.
2. Press **Insert** **`(F1)`**. You can now insert ladder logic at the cursor location.



```
|RELAY  |TMRCTR |MATH    |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1-] [- 2-]/[- 3       4       5-( )- 6-(SM)- 7-(RM)- 8vert | 9horz -10more
>
|[  START OF LD  PROGRAM LESSON    ]       (*                                *)
|
|[     VARIABLE DECLARATIONS        ]
|
|[       BLOCK DECLARATIONS         ]
|
|[     START OF PROGRAM LOGIC       ]
|
|
|
|[      END OF PROGRAM LOGIC        ]
                                     OFFLINE
C:\LM90\LESSON                      PRG: LESSON  BLK: _MAIN  SIZE:   123 RUNG 0004
REPLACE                                :           ::
```

The top line of the display shows these ladder logic functions:

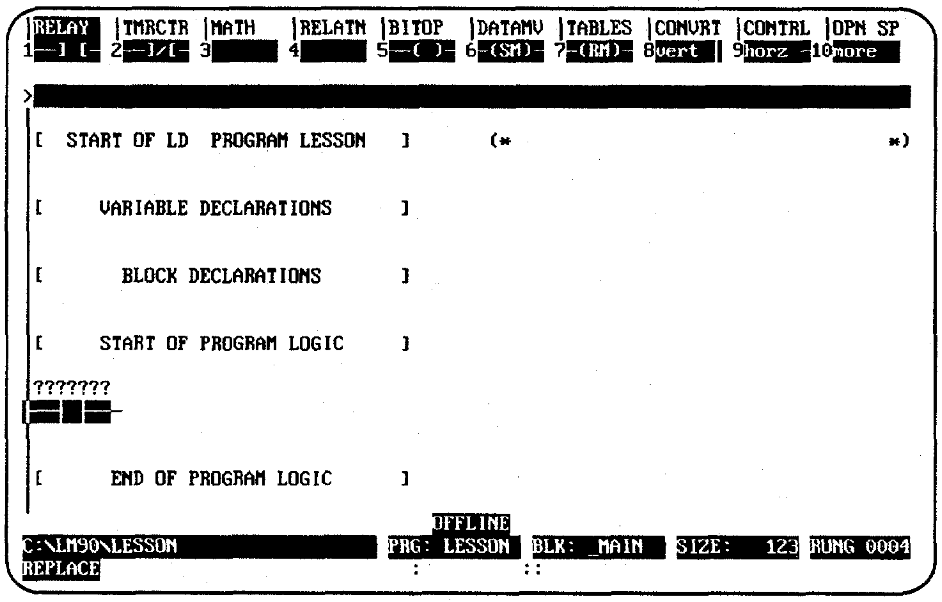
- Relay contacts and coils.
- Timers and counters.
- Math functions.
- Relational (comparison) functions.
- Logical bit operation functions.
- Data move functions.
- Table functions
- Conversion functions.
- Control functions.
- Open space operations.

All of these functions are described in chapter 3, "Program Editing." For additional information, refer to the *Series 90-30/20/Micro Programmable Controllers Reference Manual*, GFK-0467.

Note

ALT-E can be used to toggle the display of the status area at the bottom of the screen. Press **ALT-E** to remove the status information; then, press **ALT-E** again to display it again. For more information on the status area, refer to chapter 2, section 6, "Screen Format."

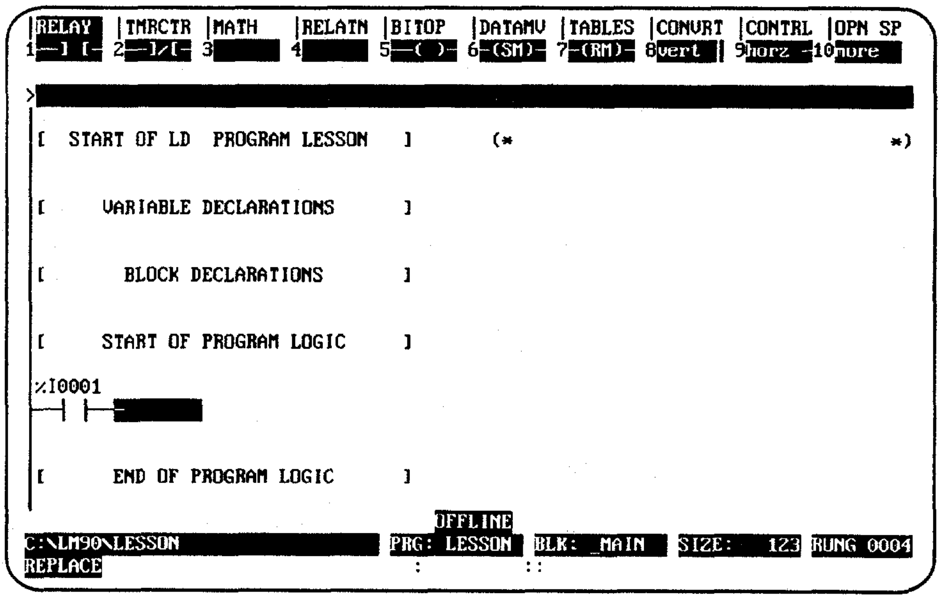
3. The reverse-video block on the top line shows that the currently selected type of function is RELAY. Press **F1** to start the rung with a normally open contact.



```
|RELAY  |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1-] [- 2-]/[- 3       4       5-( )- 6-(SM)- 7-(RM)- 8vert | 9horz -10more
>
[  START OF LD  PROGRAM LESSON    ]       (*                                    *)

[     VARIABLE DECLARATIONS       ]

[       BLOCK DECLARATIONS        ]

[     START OF PROGRAM LOGIC      ]

???????
|----

[      END OF PROGRAM LOGIC       ]
                                   OFFLINE
C:\LM90\LESSON                    PRG: LESSON  BLK: _MAIN  SIZE:    123 RUNG 0004
REPLACE                               :            ::
```

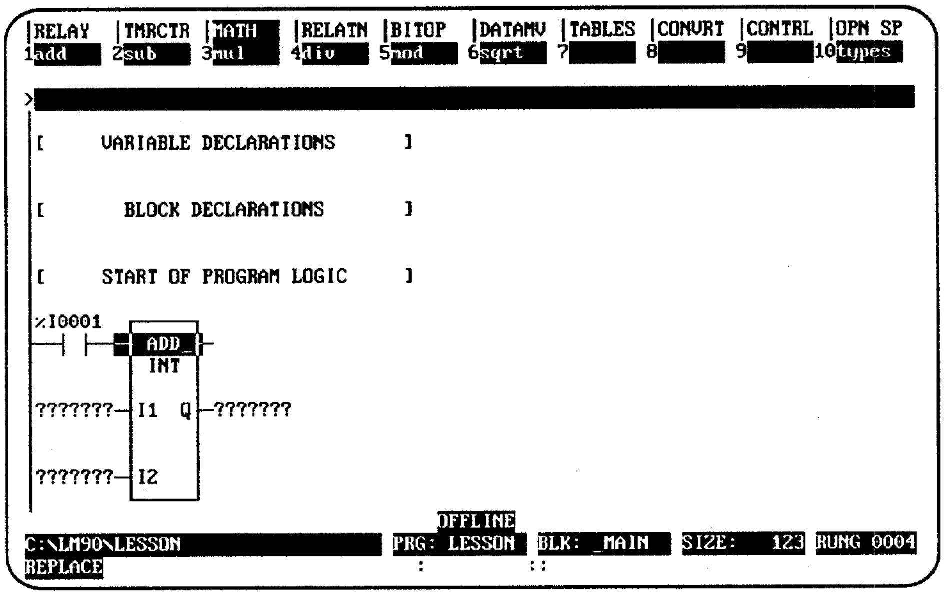
4. Type in **%I** as a reference for the contact. Note that typing **1I** has the same result. The reference appears in the command line at the top of the screen.

5. Press the **Enter** key. The software automatically changes your entry to the correct format and moves the cursor to the next location in the rung.



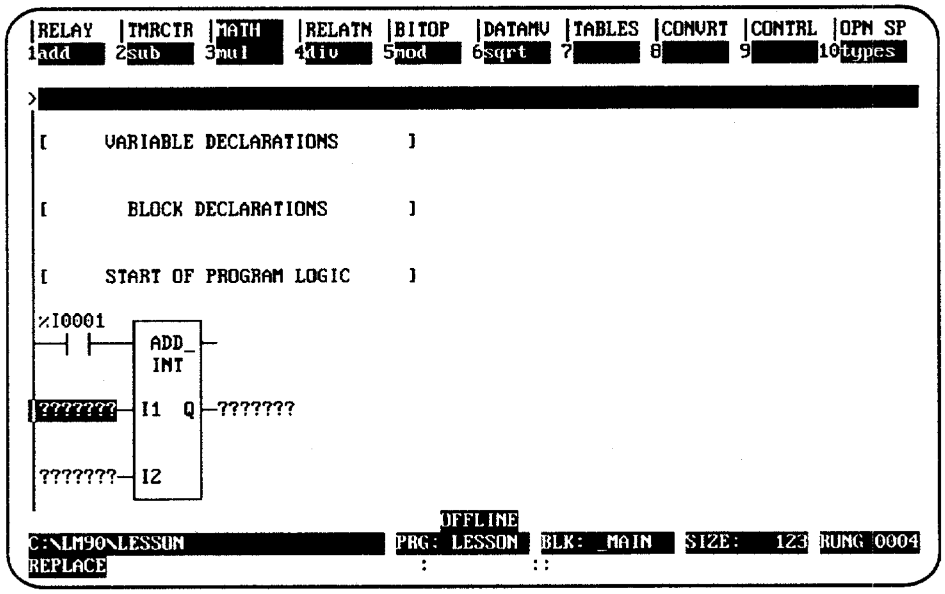
```
|RELAY  |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1-] [- 2-]/[- 3       4       5-( )- 6-(SM)- 7-(RM)- 8vert | 9horz -10more
>
[  START OF LD  PROGRAM LESSON    ]       (*                                    *)

[     VARIABLE DECLARATIONS       ]

[       BLOCK DECLARATIONS        ]

[     START OF PROGRAM LOGIC      ]

%I0001
|--| |--

[      END OF PROGRAM LOGIC       ]
                                   OFFLINE
C:\LM90\LESSON                    PRG: LESSON  BLK: _MAIN  SIZE:    123 RUNG 0004
REPLACE                               :            ::
```

6. Add a math function to the rung. Press **Math** **(Shift-F3)** to select the math functions. At the top of the screen, the reverse video block moves to MATH. The second line shows the types of math functions that are currently available. Press **Add** **(F1)** to place an addition function in the rung.



```
|RELAY  |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1add    2sub    3mul    4div    5mod    6sqrt   7       8       9       10types
>
[     VARIABLE DECLARATIONS      ]

[      BLOCK DECLARATIONS        ]

[     START OF PROGRAM LOGIC     ]
%I0001
--| |---ADD_ -
        INT
???????-I1  Q-???????

???????-I2
                                       OFFLINE
C:\LM90\LESSON                          PRG: LESSON BLK: _MAIN  SIZE:    123 RUNG 0004
REPLACE                                   :          ::
```

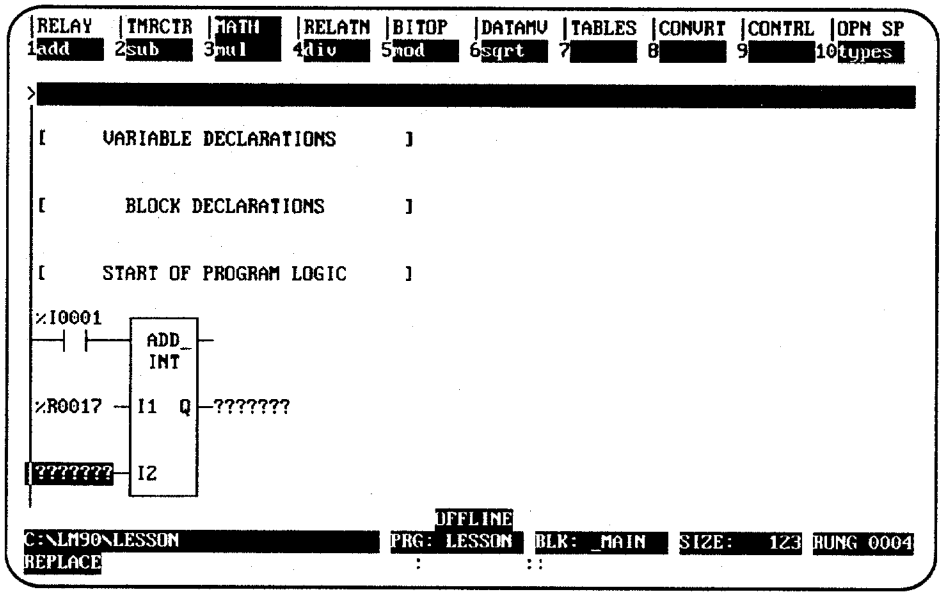
7. Use the **Tab** key to move the cursor to the first input parameter (I1). This represents the first of the two numbers to be added.



```
|RELAY  |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1add    2sub    3mul    4div    5mod    6sqrt   7       8       9       10types
>
[     VARIABLE DECLARATIONS      ]

[      BLOCK DECLARATIONS        ]

[     START OF PROGRAM LOGIC     ]
%I0001
--| |---ADD_ -
        INT
???????-I1  Q-???????

???????-I2
                                       OFFLINE
C:\LM90\LESSON                          PRG: LESSON BLK: _MAIN  SIZE:    123 RUNG 0004
REPLACE                                   :          ::
```

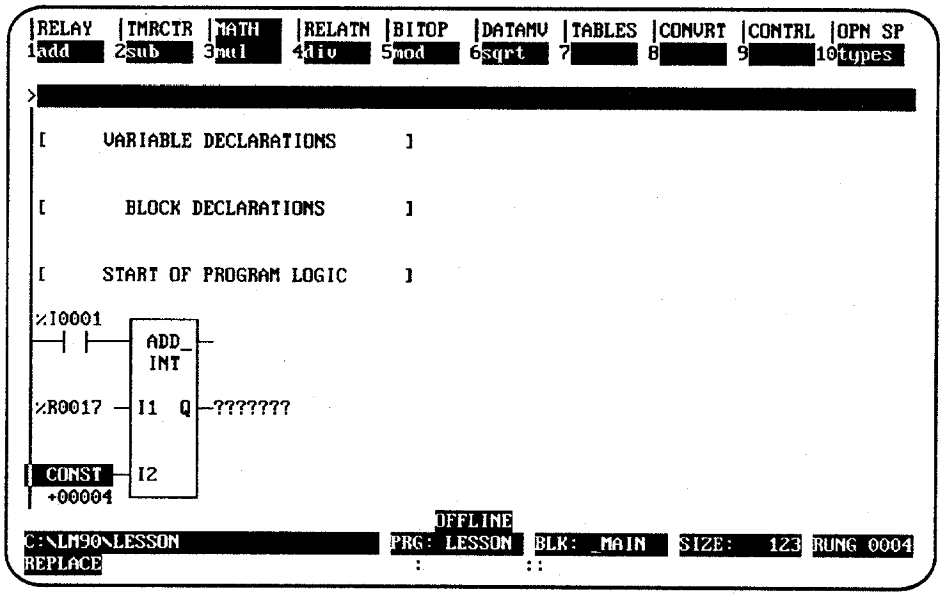
8. Specify the first number to be added. For this lesson, enter **%R17** or **17R**. This represents register reference %R0017. Then, press the **Enter** key.

9. Use the **Tab** key to move the cursor to the second input parameter (I2). This represents the number to be added to the current value of the reference %R0017.



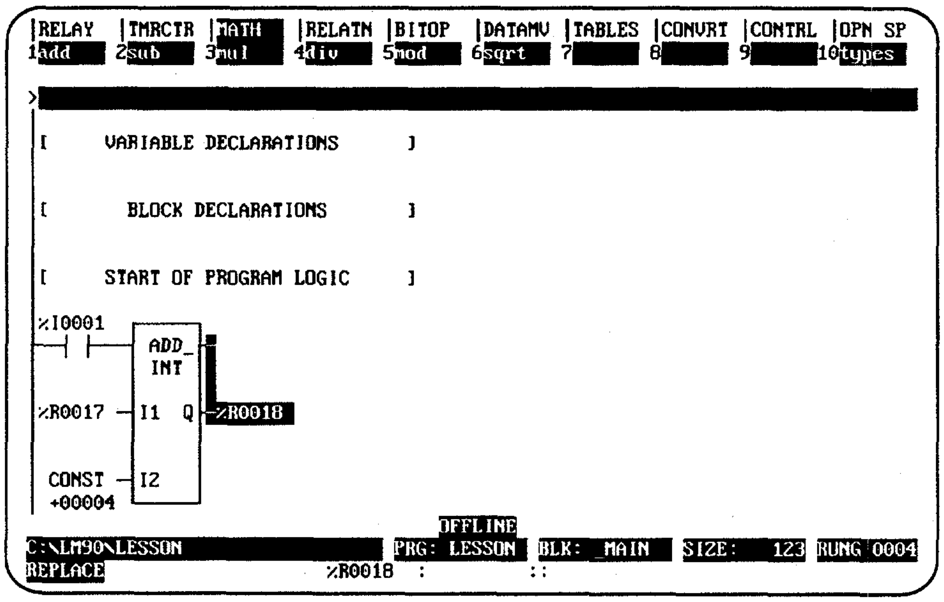
```
|RELAY  |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1add    2sub    3mul    4div    5mod    6sqrt   7       8       9       10types
>
[     VARIABLE DECLARATIONS       ]

[       BLOCK DECLARATIONS        ]

[     START OF PROGRAM LOGIC      ]
%I0001
--| |----ADD_ -
         INT

%R0017 - I1  Q -???????

???????- I2

                                   OFFLINE
C:\LM90\LESSON                    PRG: LESSON  BLK: _MAIN  SIZE:   123 RUNG 0004
REPLACE                                :          ::
```

10. Enter the number **4** as the value to be added. Press the **Enter** key. Because you did not type a % symbol before the number, the software knows it is a constant.



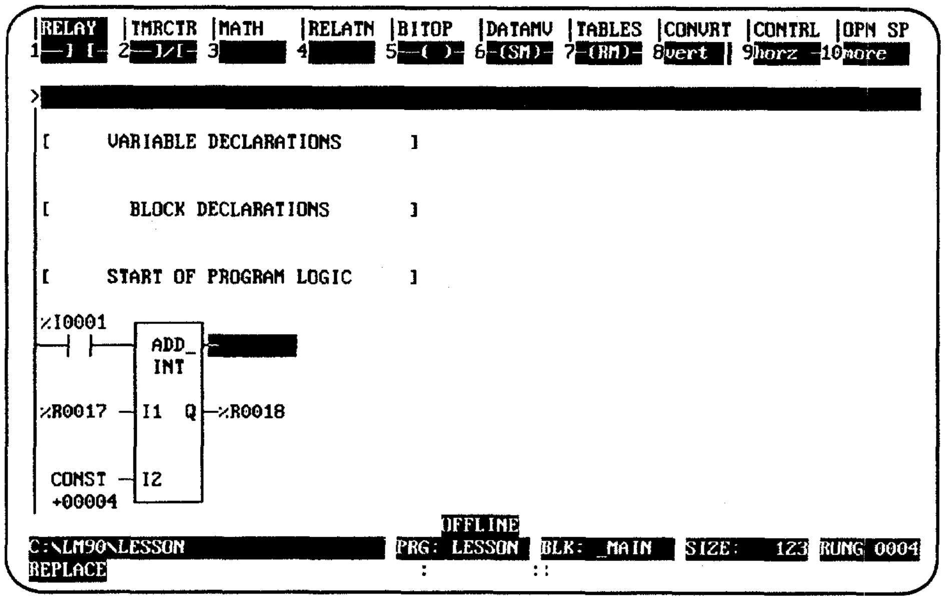
```
|RELAY  |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1add    2sub    3mul    4div    5mod    6sqrt   7       8       9       10types
>
[     VARIABLE DECLARATIONS       ]

[       BLOCK DECLARATIONS        ]

[     START OF PROGRAM LOGIC      ]
%I0001
--| |----ADD_ -
         INT

%R0017 - I1  Q -???????

 CONST - I2
 +00004
                                   OFFLINE
C:\LM90\LESSON                    PRG: LESSON  BLK: _MAIN  SIZE:   123 RUNG 0004
REPLACE                                :          ::
```

11. Use the **Tab** key to move the cursor to the next parameter (Q). Q is an output reference which represents the location for the sum (output) of the addition. Enter the reference `%R18`, and press the **Enter** key.



```
|RELAY  |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1add    2sub    3mul    4div    5mod    6sqrt   7       8       9       10types
>
[     VARIABLE DECLARATIONS      ]

[       BLOCK DECLARATIONS       ]

[     START OF PROGRAM LOGIC     ]

%I0001
--| |---ADD_
        INT
%R0017 -I1  Q-%R0018

 CONST -I2
 +00004
                                        OFFLINE
C:\LM90\LESSON                          PRG: LESSON  BLK: _MAIN  SIZE:   123 RUNG 0004
REPLACE                   %R0018  :         ::
```

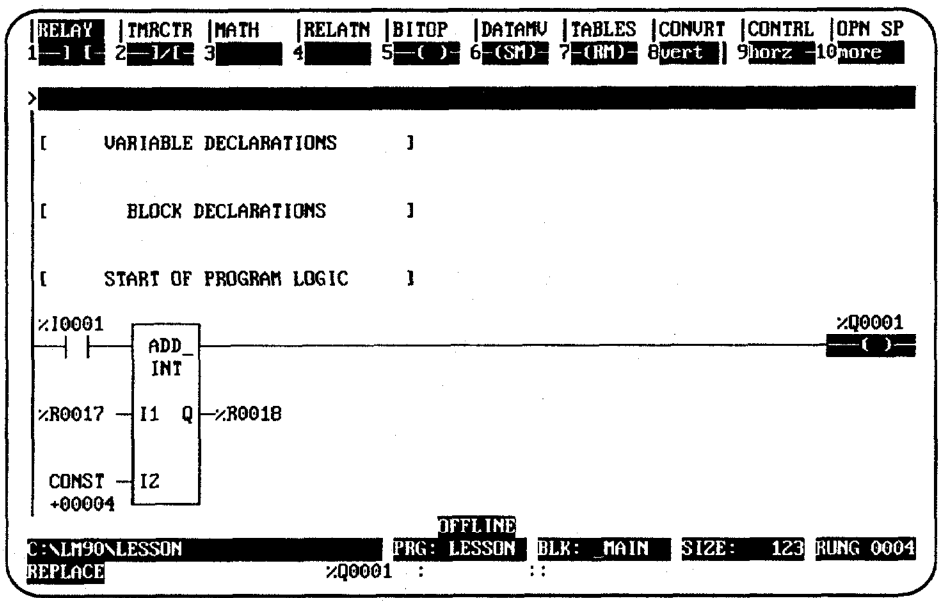
12. Use the **Tab** key to move the cursor block up to the top rung line. To complete the rung by programming a coil, press **Shift-F1** to select the relay functions.



```
|RELAY  |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1-] [-  2-]/[-  3       4       5-( )-  6-(SM)- 7-(RM)- 8vert | 9horz - 10more
>
[     VARIABLE DECLARATIONS      ]

[       BLOCK DECLARATIONS       ]

[     START OF PROGRAM LOGIC     ]

%I0001
--| |---ADD_
        INT
%R0017 -I1  Q-%R0018

 CONST -I2
 +00004
                                        OFFLINE
C:\LM90\LESSON                          PRG: LESSON  BLK: _MAIN  SIZE:   123 RUNG 0004
REPLACE                           :         ::
```

13. Enter `%Q1` into the command line. This will be the reference for the coil.

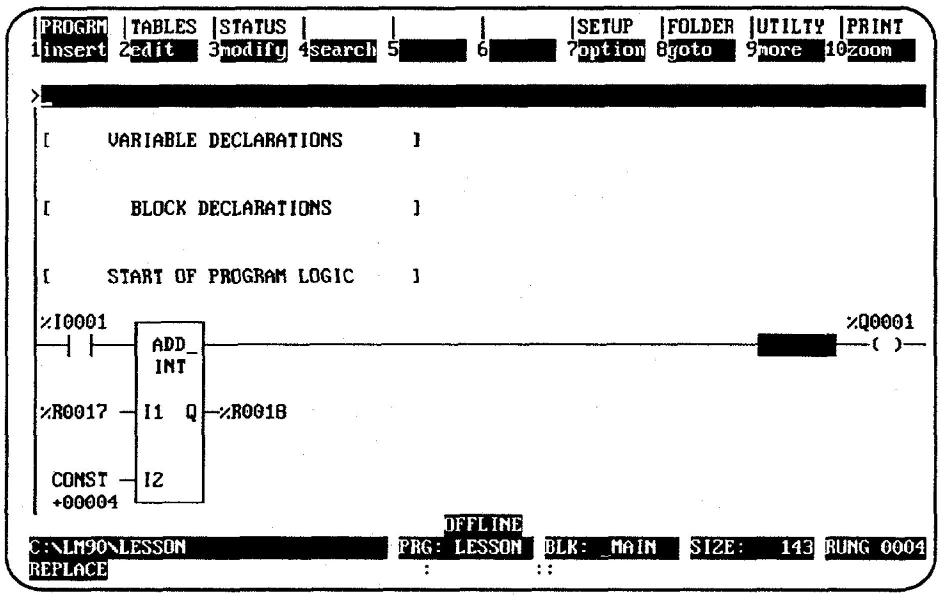
14. Press **F5** to enter a coil at the end of the rung. Notice that the reference can be entered either before selecting the function as shown here, or after selecting the function as shown earlier.



```
|RELAY  |TMRCTR |MATH    |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1-] [-  2-]/[-  3        4       5-( )-  6-(SM)- 7-(RM)- 8vert | 9horz - 10more
>
[     VARIABLE DECLARATIONS      ]

[       BLOCK DECLARATIONS       ]

[     START OF PROGRAM LOGIC     ]

%I0001                                                              %Q0001
--| |---| ADD_ |------------------------------------------------------( )-
        | INT  |

%R0017 -|I1  Q|-%R0018

 CONST -|I2    |
 +00004
                                          OFFLINE
C:\LM90\LESSON                            PRG: LESSON  BLK: _MAIN  SIZE:   123 RUNG 0004
REPLACE                         %Q0001  :          ::
```

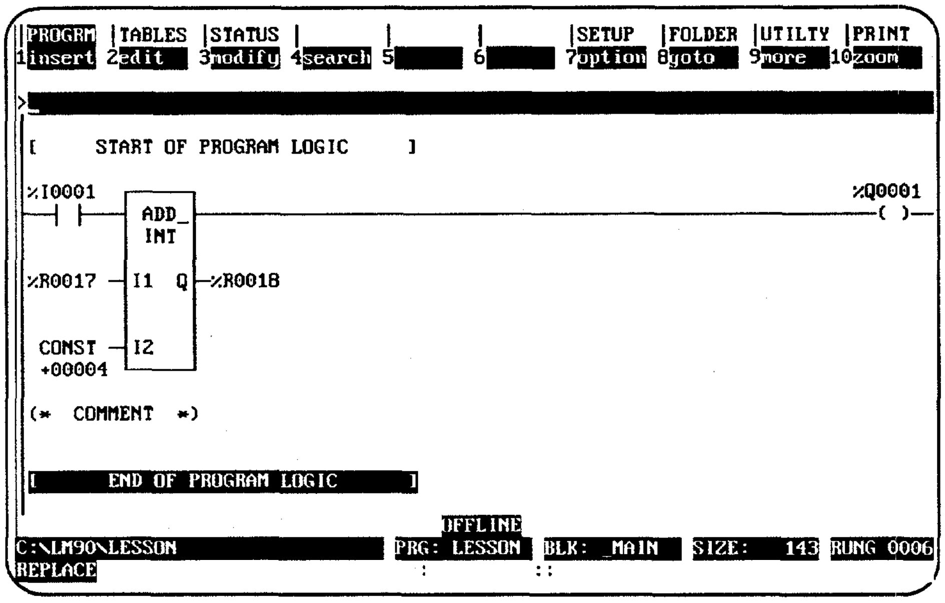
15. To accept the rung, press the **Enter** key (or the Plus (+) key on the numeric keypad). This accepts the completed rung, and the cursor moves downward so you can insert another line of logic.

16. Select the control functions by pressing **Shift-F9**. Press **Comment (F8)** to add a comment to the program.



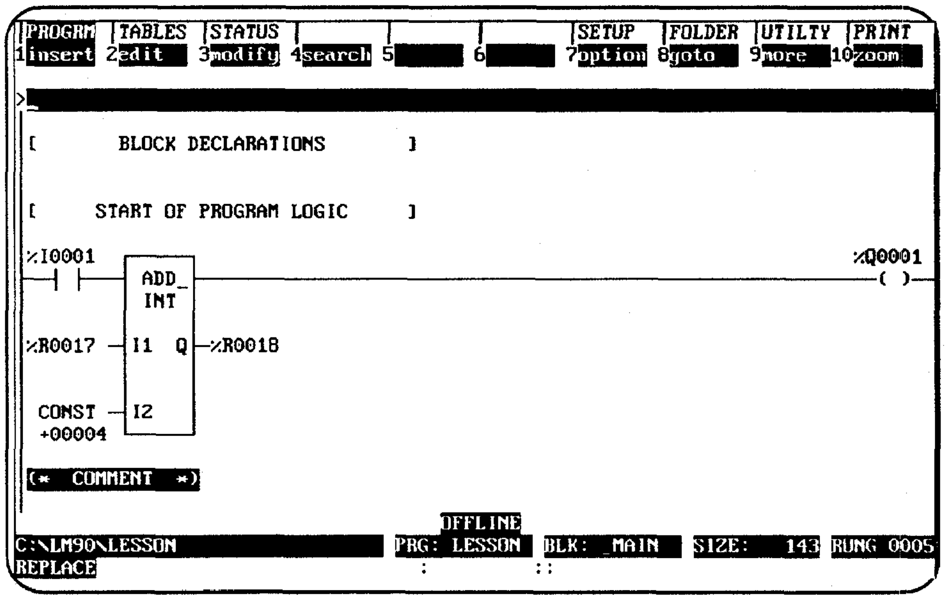
```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3modify 4search 5       6       7option 8goto   9more   10zoom
>_
[     VARIABLE DECLARATIONS      ]

[       BLOCK DECLARATIONS       ]

[     START OF PROGRAM LOGIC     ]

%I0001                                                              %Q0001
--| |---| ADD_ |----------------------------------------------------------( )-
        | INT  |

%R0017 -|I1  Q|-%R0018

 CONST -|I2    |
 +00004
                                          OFFLINE
C:\LM90\LESSON                            PRG: LESSON  BLK: _MAIN  SIZE:   143 RUNG 0004
REPLACE                                 :          ::
```

17. Press the **Enter** key and then the **Escape** key to return these key functions to the top of the screen as shown below:



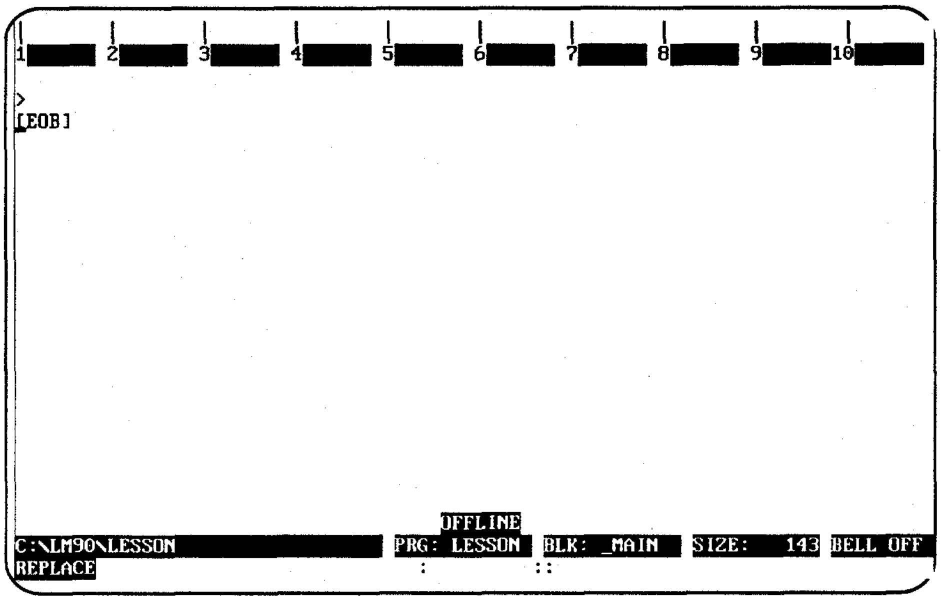
```
|PROGRM |TABLES |STATUS |        |        |        |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3modify 4search 5        6        7option 8goto   9more  10zoom
>_

[     START OF PROGRAM LOGIC      ]

%I0001                                                              %Q0001
--| |----ADD_ ----------------------------------------------------------( )--
          INT

%R0017 --I1   Q--%R0018

 CONST --I2
 +00004

(*  COMMENT  *)

[     END OF PROGRAM LOGIC        ]
                                      OFFLINE
C:\LM90\LESSON                        PRG: LESSON  BLK: _MAIN  SIZE:   143  RUNG 0006
REPLACE                                 :           ::
```

18. Press the **Up Cursor Movement** (or **Up Arrow)** key to move the cursor up to highlight the COMMENT instruction as shown below:



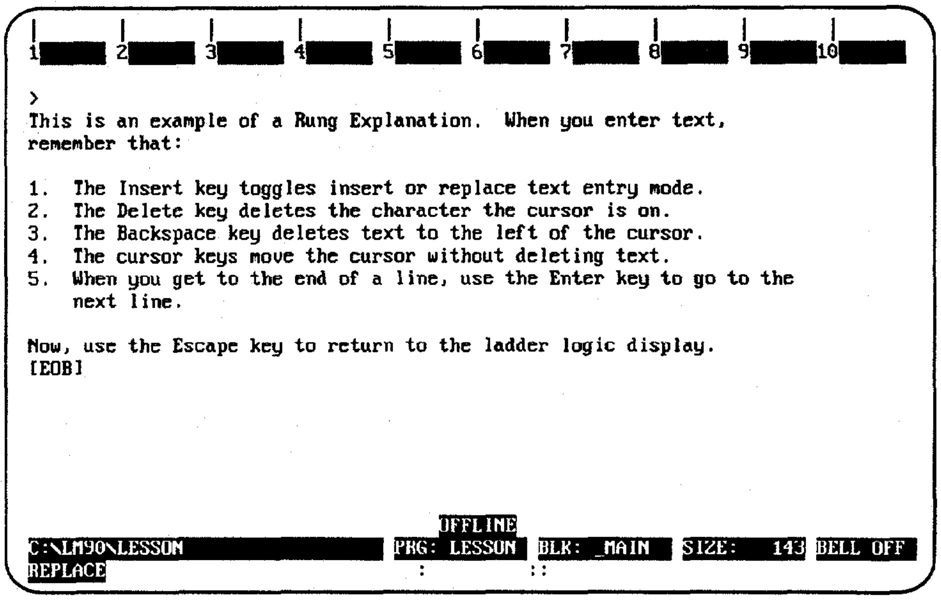
```
|PROGRM |TABLES |STATUS |        |        |        |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3modify 4search 5        6        7option 8goto   9more  10zoom
>_

[        BLOCK DECLARATIONS       ]

[     START OF PROGRAM LOGIC      ]

%I0001                                                              %Q0001
--| |----ADD_ ----------------------------------------------------------( )--
          INT

%R0017 --I1   Q--%R0018

 CONST --I2
 +00004

(*  COMMENT  *)
                                      OFFLINE
C:\LM90\LESSON                        PRG: LESSON  BLK: _MAIN  SIZE:   143  RUNG 0005
REPLACE                                 :           ::
```

19. Press **Zoom** (F10). The program logic is replaced with a blank screen where you can enter text.



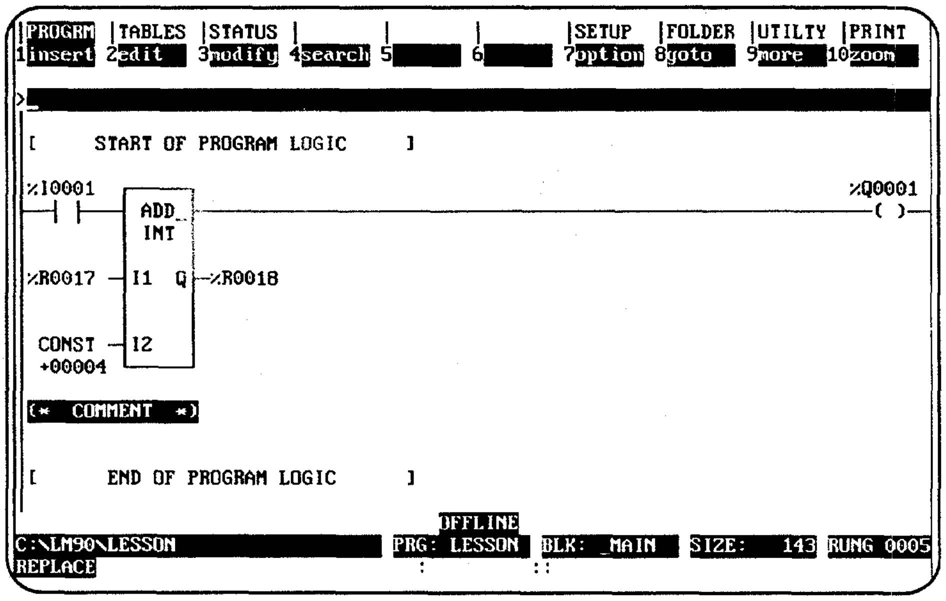
```
 1      2      3      4      5      6      7      8      9      10
>
[EOB]




                                        OFFLINE
C:\LM90\LESSON                          PRG: LESSON  BLK: _MAIN  SIZE:   143 BELL OFF
REPLACE                                   :          ::
```

20. Beginning at the **[EOB]** (End of Buffer) symbol, enter the text shown below:



```
 1      2      3      4      5      6      7      8      9      10
>
This is an example of a Rung Explanation.  When you enter text,
remember that:

1.  The Insert key toggles insert or replace text entry mode.
2.  The Delete key deletes the character the cursor is on.
3.  The Backspace key deletes text to the left of the cursor.
4.  The cursor keys move the cursor without deleting text.
5.  When you get to the end of a line, use the Enter key to go to the
    next line.

Now, use the Escape key to return to the ladder logic display.
[EOB]

                                        OFFLINE
C:\LM90\LESSON                          PRG: LESSON  BLK: _MAIN  SIZE:   143 BELL OFF
REPLACE                                   :          ::
```

21. Press the **Escape** key to save the comment and return the ladder logic display to the screen.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3modify 4search 5       6       7option 8goto   9more   10zoom
>_
[     START OF PROGRAM LOGIC      ]

%I0001                                                            %Q0001
--| |---[ ADD_ ]---------------------------------------------------( )--
         INT
%R0017 -|I1  Q|-%R0018

 CONST -|I2
 +00004
(*  COMMENT  *)

[      END OF PROGRAM LOGIC       ]
                                   OFFLINE
C:\LM90\LESSON                     PRG: LESSON  BLK: _MAIN  SIZE:   143 RUNG 0005
REPLACE
```

22. You could display the comment again by pressing **Zoom** `(F10)` with the cursor located at the comment rung.

23. That completes the lesson. Press the **Escape** key to return to the main menu. The following information pertains to subroutines. If you want to practice creating a subroutine, continue below. If you want to print this program now, continue on page A-23.

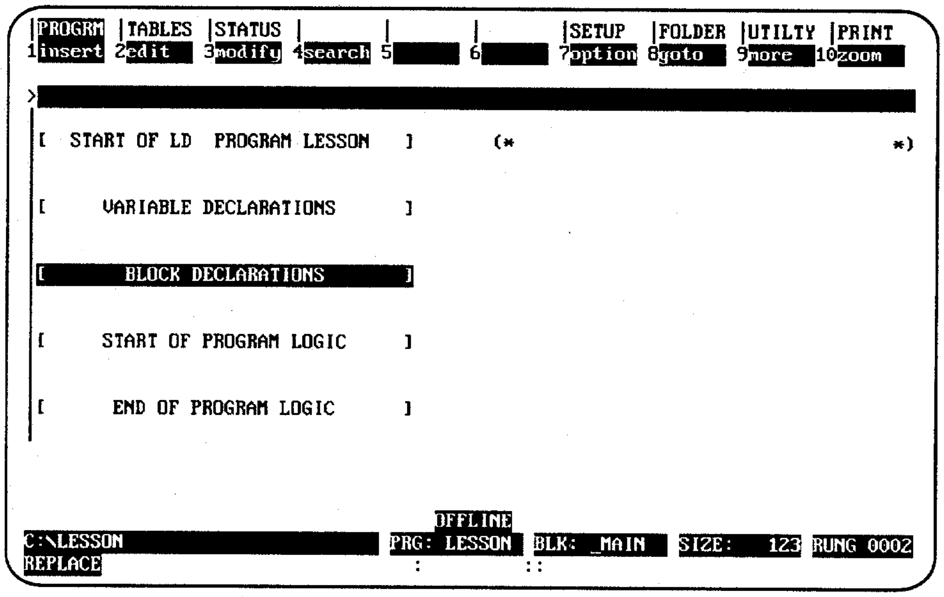
Creating a Subroutine Block

Subroutines are declared through the block declaration editor.

Note

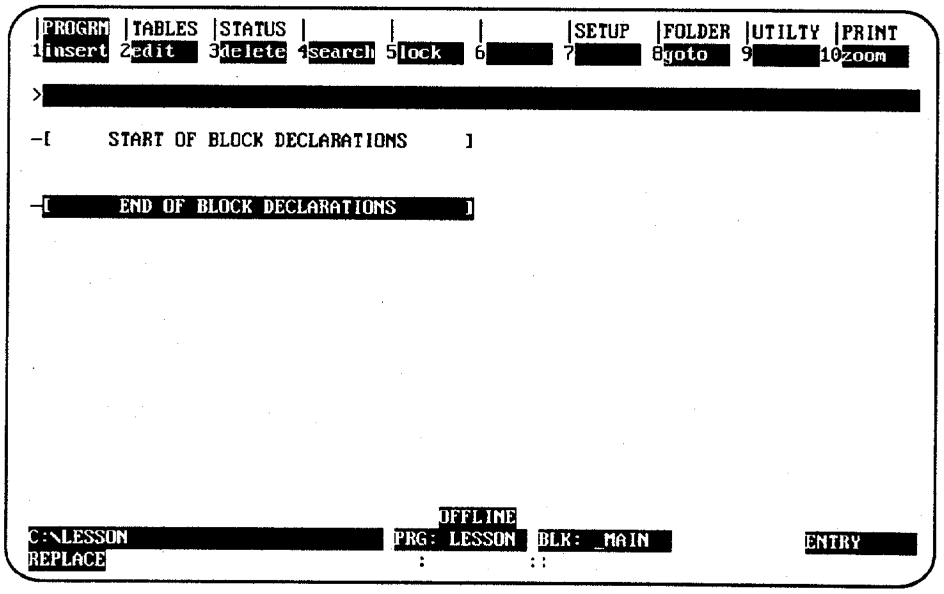
Subroutine blocks are ***not*** available for the Series 90-20 PLC nor for Micro PLCs.

1. To create a subroutine declaration, position the cursor on the **`[BLOCK DECLARATIONS]`** marker.



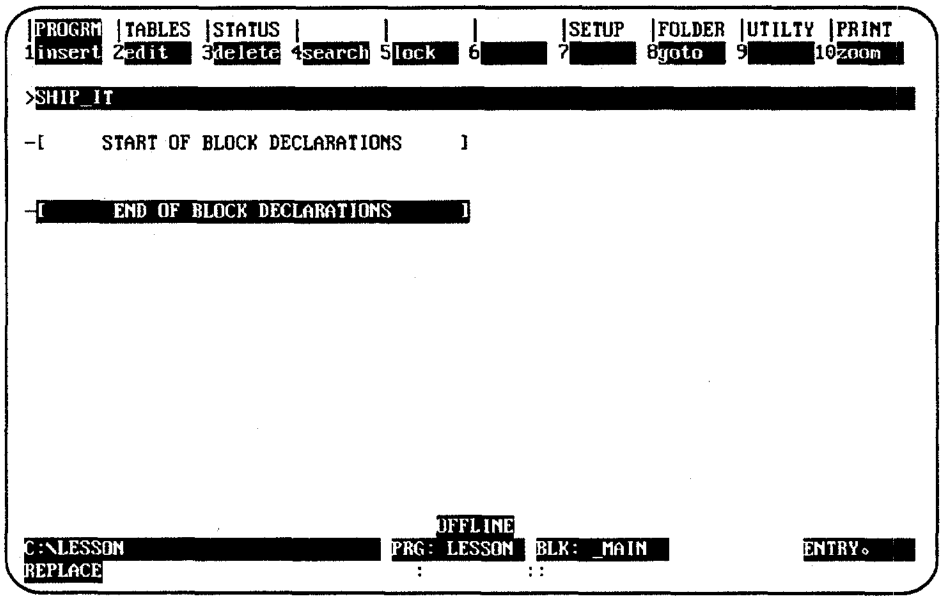
```
|PROGRM |TABLES |STATUS |        |        |        |SETUP   |FOLDER  |UTILTY  |PRINT
1insert 2edit   3modify 4search 5        6        7option  8goto    9more   10zoom
>
[  START OF LD  PROGRAM LESSON   ]        (*                                        *)

[     VARIABLE DECLARATIONS      ]

[       BLOCK DECLARATIONS       ]

[     START OF PROGRAM LOGIC     ]

[      END OF PROGRAM LOGIC      ]

                                          OFFLINE
C:\LESSON                                 PRG: LESSON  BLK: _MAIN   SIZE:    123  RUNG 0002
REPLACE                                        :             ::
```

2. Then, press the **Zoom** **`(F10)`** key.



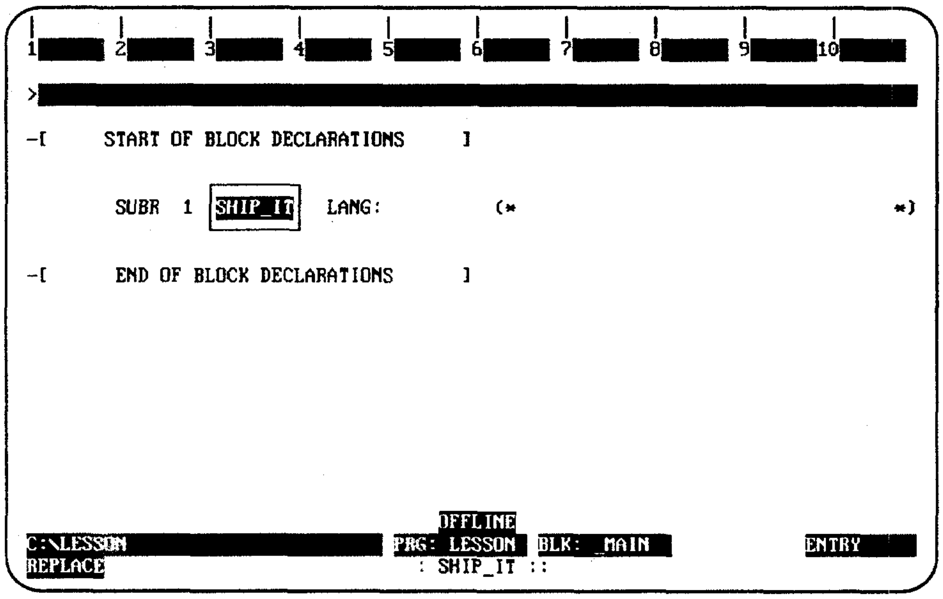
```
|PROGRM |TABLES |STATUS |        |        |        |SETUP   |FOLDER  |UTILTY  |PRINT
1insert 2edit   3delete 4search 5lock    6        7        8goto    9       10zoom
>
-[     START OF BLOCK DECLARATIONS      ]

-[      END OF BLOCK DECLARATIONS       ]

                                          OFFLINE
C:\LESSON                                 PRG: LESSON  BLK: _MAIN                 ENTRY
REPLACE                                        :             ::
```

3. Enter the name of the subroutine on the command line. For this lesson, type in the name **SHIP_IT**



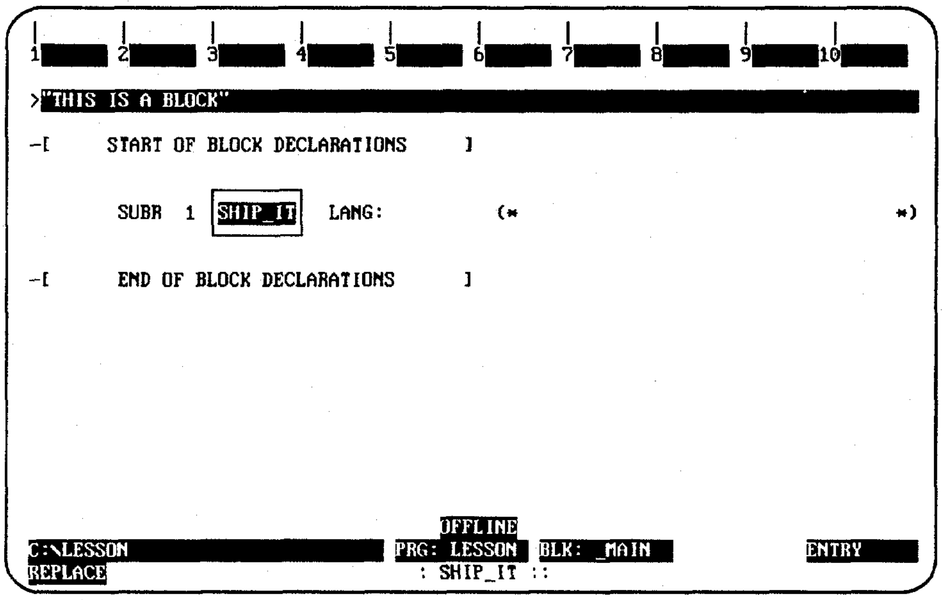
```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3delete 4search 5lock   6       7       8goto   9       10zoom
>SHIP_IT

-[      START OF BLOCK DECLARATIONS      ]


-[       END OF BLOCK DECLARATIONS       ]









                                              OFFLINE
C:\LESSON                                  PRG: LESSON  BLK: _MAIN            ENTRY◦
REPLACE                                      :            ::
```

4. Then, press the **Insert** **(F1)** key.



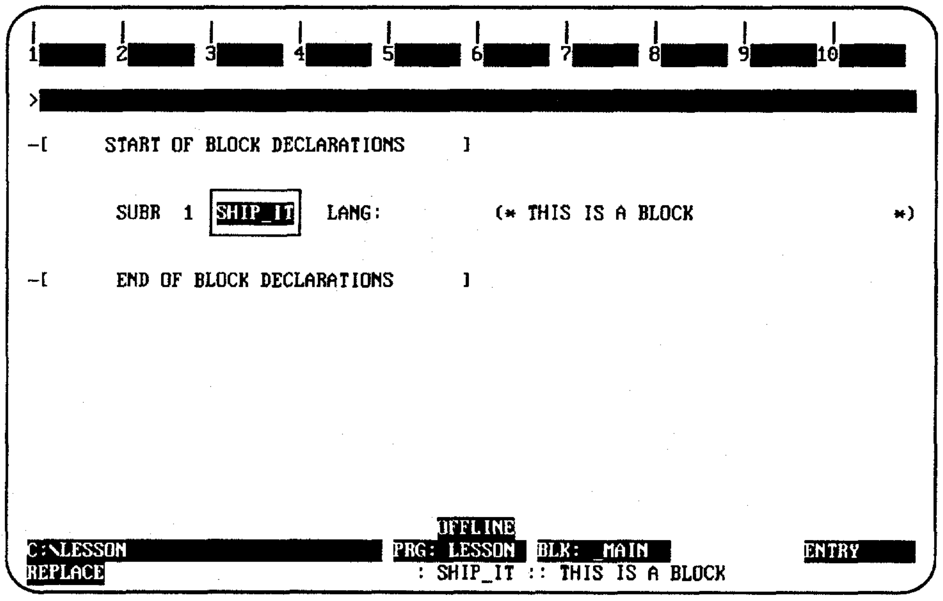
```
|       |       |       |       |       |       |       |       |       |
1       2       3       4       5       6       7       8       9       10
>

-[      START OF BLOCK DECLARATIONS      ]

        SUBR  1  SHIP_IT   LANG:         (*                                *)

-[      END OF BLOCK DECLARATIONS       ]









                                              OFFLINE
C:\LESSON                                  PRG: LESSON  BLK: _MAIN            ENTRY
REPLACE                                      : SHIP_IT ::
```

5. An explanation of up to 32 characters can also be entered at this time. Type the description **"THIS IS A BLOCK"** on the command line.



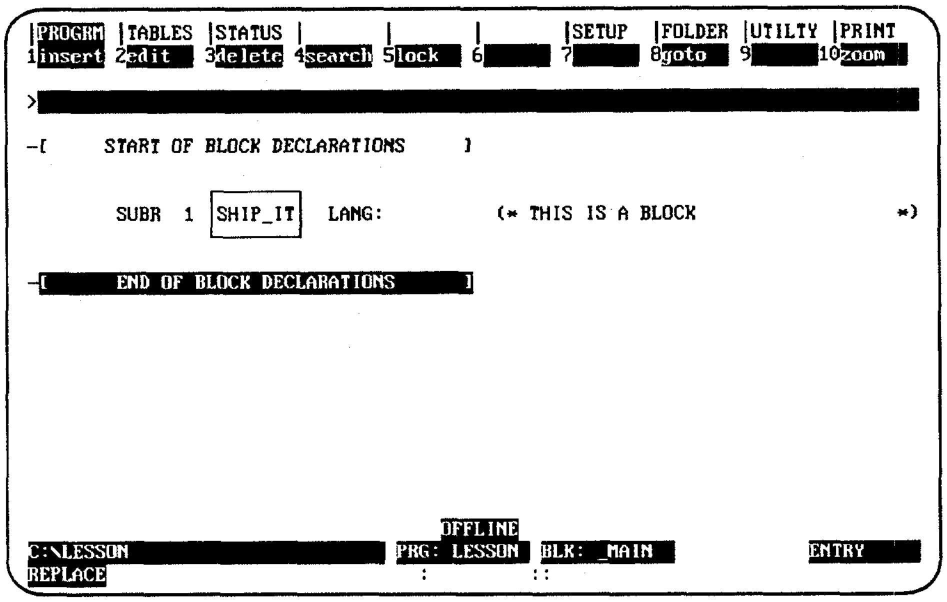
```
 1      2      3      4      5      6      7      8      9      10
>"THIS IS A BLOCK"
-[     START OF BLOCK DECLARATIONS      ]

        SUBR  1  SHIP_IT  LANG:          (*                          *)

-[      END OF BLOCK DECLARATIONS       ]

                                         OFFLINE
C:\LESSON                               PRG: LESSON  BLK: _MAIN          ENTRY
REPLACE                                   : SHIP_IT ::
```

6. Press the **Enter** key.



```
 1      2      3      4      5      6      7      8      9      10
>
-[     START OF BLOCK DECLARATIONS      ]

        SUBR  1  SHIP_IT  LANG:          (* THIS IS A BLOCK          *)

-[      END OF BLOCK DECLARATIONS       ]

                                         OFFLINE
C:\LESSON                               PRG: LESSON  BLK: _MAIN          ENTRY
REPLACE                                   : SHIP_IT :: THIS IS A BLOCK
```

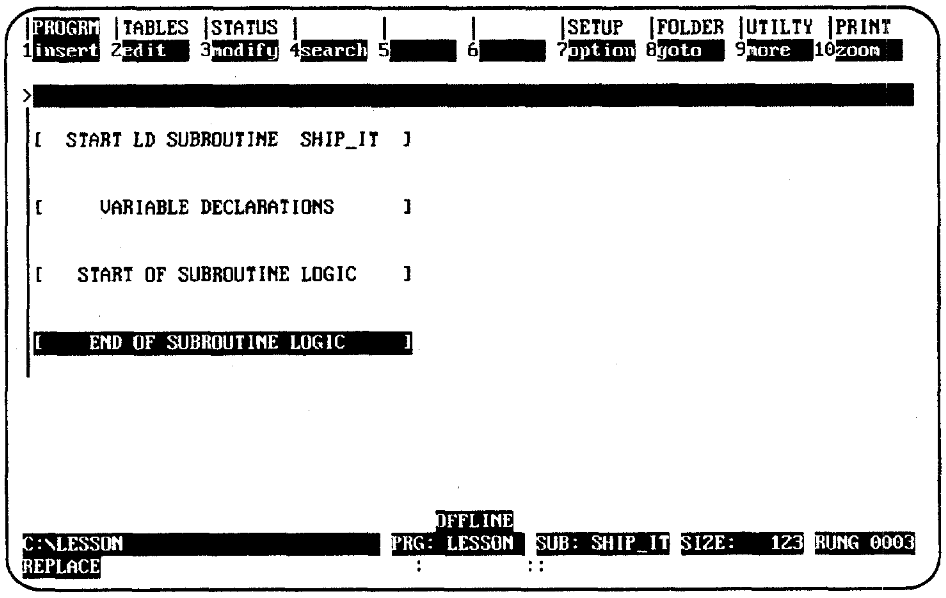
7. Then, press the **Escape** key.



```
|PROGRM |TABLES |STATUS |        |        |        |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3delete 4search 5lock   6       7       8goto   9       10zoom
>
-[     START OF BLOCK DECLARATIONS      ]

        SUBR  1  SHIP_IT  LANG:          (* THIS IS A BLOCK                      *)

-[       END OF BLOCK DECLARATIONS       ]

                                        OFFLINE
C:\LESSON                              PRG: LESSON  BLK: _MAIN           ENTRY
REPLACE                                   :          ::
```

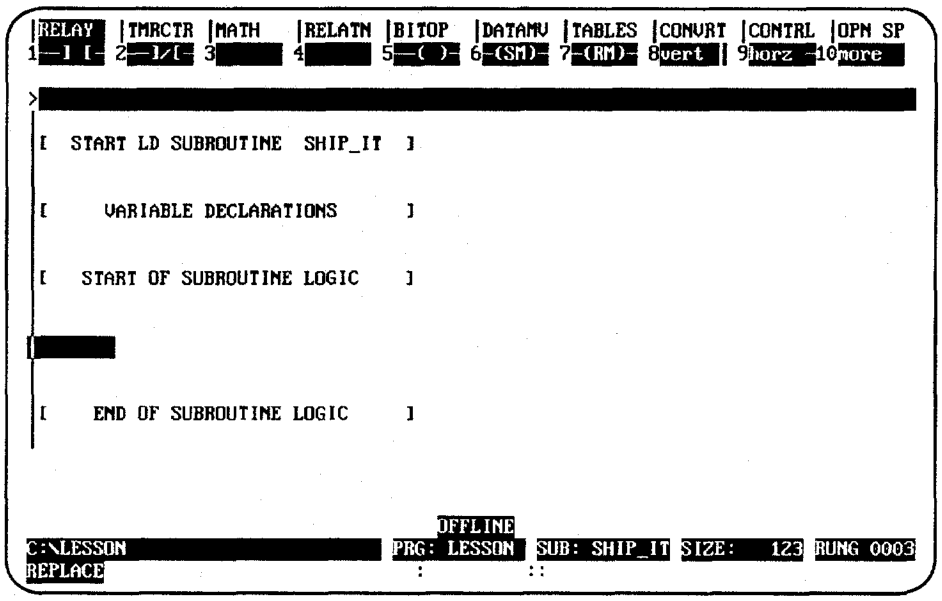
There are two alternate methods of entering this information. Both methods combine the previous step and this step into one operation. The first method is to type the name **SHIP_IT** and the description **"THIS IS A BLOCK"** together on the command line. Press the **Insert (F1)** key, and then press the Escape key. The second method is to first press the **Insert (F1)** key, type the name and description together on the command line. Then, press the **Enter** key, followed by the **Escape** key.

8. Use the cursor keys to position the cursor on the subroutine block declaration named **SHIP_IT** Then, press **Zoom (F10)**. Note that the name of the block (**BLK: _MAIN**) in the status area of the previous screen has now been changed to the name of the subroutine block (**SUB: SHIP_IT**).



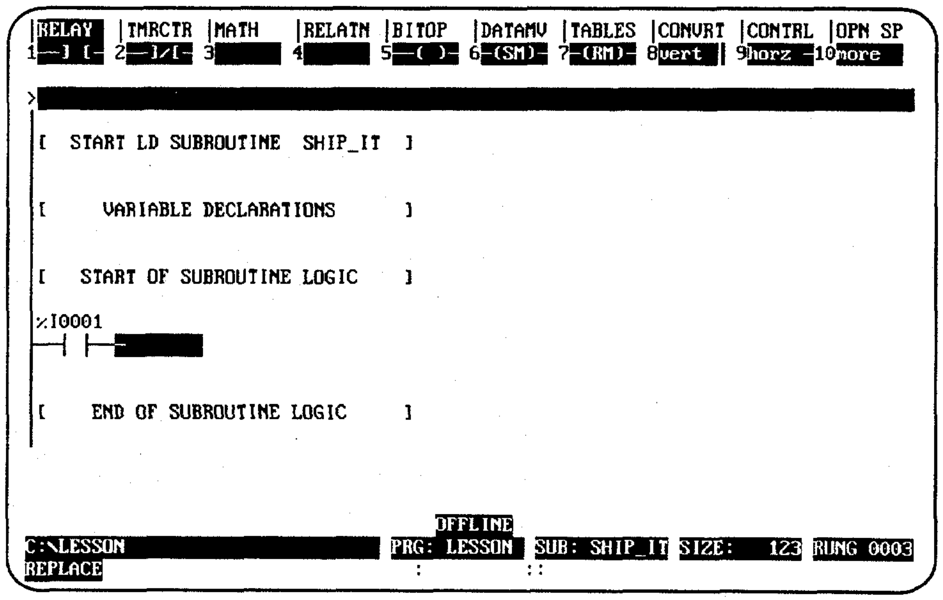
```
|PROGRM |TABLES |STATUS |        |        |        |SETUP  |FOLDER |UTILTY |PRINT
1insert 2edit   3modify 4search 5       6       7option 8goto   9more   10zoom
>
[  START LD SUBROUTINE  SHIP_IT  ]

[     VARIABLE DECLARATIONS      ]

[   START OF SUBROUTINE LOGIC    ]

[    END OF SUBROUTINE LOGIC     ]

                                        OFFLINE
C:\LESSON                              PRG: LESSON  SUB: SHIP_IT SIZE:    123 RUNG 0003
REPLACE                                   :          ::
```

9. You can create logic for the subroutine block on this screen. Position the cursor on the **[END OF SUBROUTINE LOGIC]** marker and press **Insert (F1)**. You can now insert ladder logic at the cursor location.



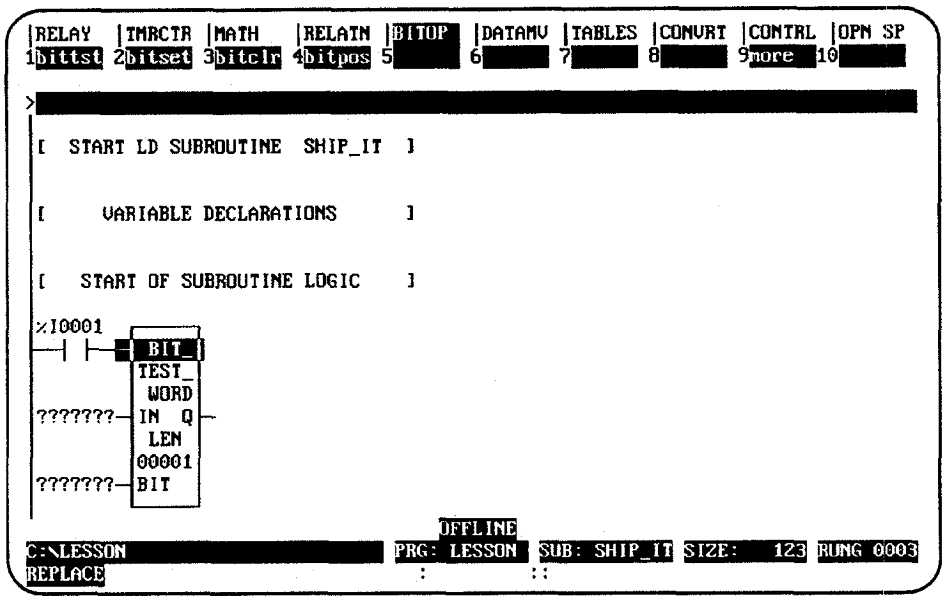
```
|RELAY |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1-] [- 2-]/[- 3       4       5-( )- 6-(SM)- 7-(RM)- 8vert | 9horz -10more
>
[  START LD SUBROUTINE  SHIP_IT  ]

[     VARIABLE DECLARATIONS      ]

[   START OF SUBROUTINE LOGIC    ]


[    END OF SUBROUTINE LOGIC     ]

                                           OFFLINE
C:\LESSON                                 PRG: LESSON  SUB: SHIP_IT SIZE:    123 RUNG 0003
REPLACE                                    :        ::
```

10. Press **F1** to start the rung with a normally open contact. Type in the reference **%I1**, and press the **Enter** key. The software automatically changes your entry to the correct format and moves the cursor to the next location in the rung.



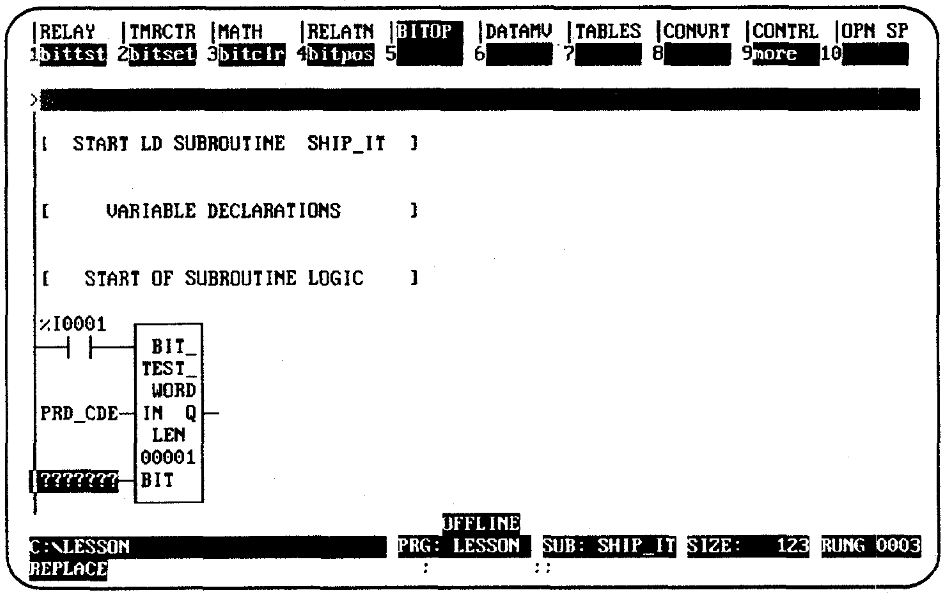
```
|RELAY |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1-] [- 2-]/[- 3       4       5-( )- 6-(SM)- 7-(RM)- 8vert | 9horz -10more
>
[  START LD SUBROUTINE  SHIP_IT  ]

[     VARIABLE DECLARATIONS      ]

[   START OF SUBROUTINE LOGIC    ]

%I0001
--| |--

[    END OF SUBROUTINE LOGIC     ]

                                           OFFLINE
C:\LESSON                                 PRG: LESSON  SUB: SHIP_IT SIZE:    123 RUNG 0003
REPLACE                                    :        ::
```

11. Add a bit operation function to the rung. Press **Bit Operation** **(Shift-F5)** to select the functions. At the top of the screen, the reverse video block moves to BITOP. The second line shows the types of bit operation functions that are currently available. Press **More** **(F9)** to display additional bit operation functions you can select. Then, press **Bit Test** **(F1)** to insert a function in the rung that will test a bit within a bit string.



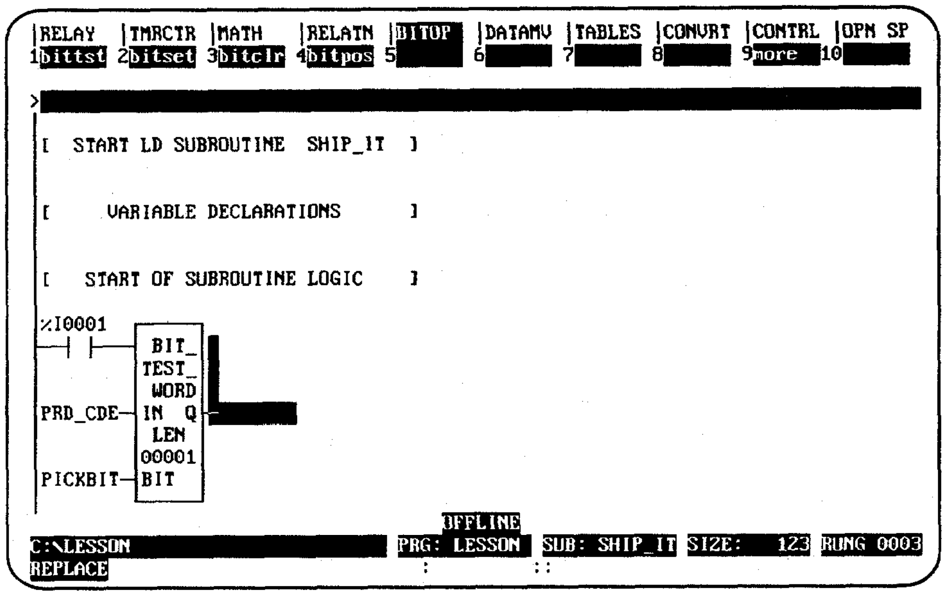
```
|RELAY  |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1bittst 2bitset 3bitclr 4bitpos 5       6       7       8       9more   10
>
[  START LD SUBROUTINE  SHIP_IT  ]

[     VARIABLE DECLARATIONS      ]

[   START OF SUBROUTINE LOGIC    ]

%I0001
--| |---| BIT_ |
        | TEST_|
        |  WORD|
???????-|IN   Q|-
        |  LEN |
        |00001 |
???????-|BIT   |
                                          OFFLINE
C:\LESSON                          PRG: LESSON  SUB: SHIP_IT SIZE:    123 RUNG 0003
REPLACE                                :          ::
```

12. Use the **Tab** key to move the cursor to the parameter IN. IN contains the first word of the data to be operated on. For this lesson, enter **PRD_CDE** on the command line. Because you have not previously entered the nickname PRD_CDE in the variable declaration table, you will also need to enter a reference address for the nickname now. After typing **PRD_CDE** on the command line, type **%R1** for the reference address. Then, press the **Tab** key. The software will enter the reference address in the logic and automatically move the cursor to the next parameter (BIT).



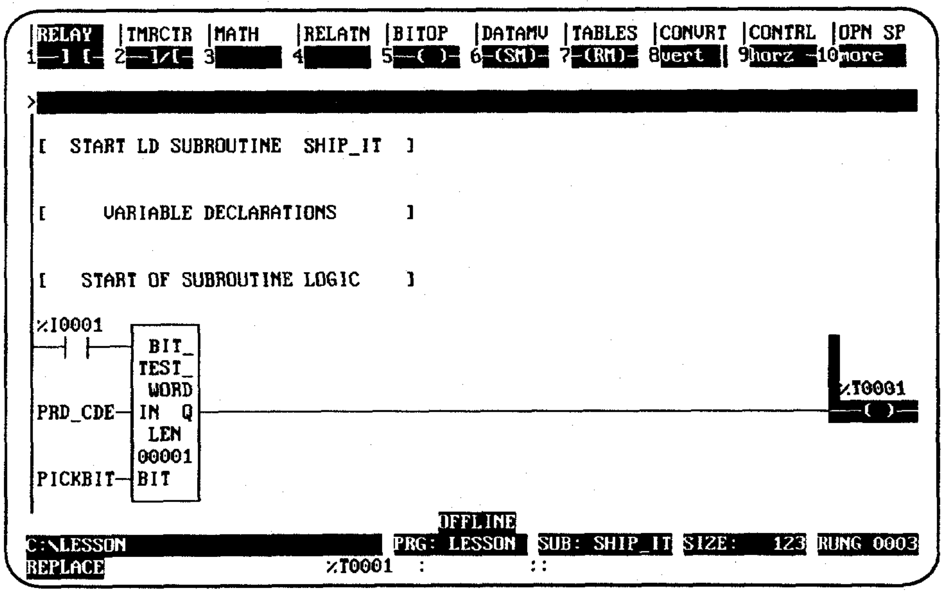
```
|RELAY  |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1bittst 2bitset 3bitclr 4bitpos 5       6       7       8       9more   10
>
[  START LD SUBROUTINE  SHIP_IT  ]

[     VARIABLE DECLARATIONS      ]

[   START OF SUBROUTINE LOGIC    ]

%I0001
--| |---| BIT_ |
        | TEST_|
        |  WORD|
PRD_CDE-|IN   Q|-
        |  LEN |
        |00001 |
???????-|BIT   |
                                          OFFLINE
C:\LESSON                          PRG: LESSON  SUB: SHIP_IT SIZE:    123 RUNG 0003
REPLACE                                :          ::
```

13. BIT contains the bit number of IN that should be tested. For this lesson, enter **PICKBIT** and the reference address **%R2** on the command line. (If the nickname PICKBIT had previously been entered in the variable declaration table, you would not need to enter it here on the command line.) Then, press the **Tab** key. Again, the software will enter the reference address in the logic and automatically move the cursor to the next parameter (Q).



```
|RELAY  |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1bittst 2bitset 3bitclr 4bitpos 5       6       7       8       9more  10
>

[  START LD SUBROUTINE  SHIP_IT  ]

[     VARIABLE DECLARATIONS      ]

[   START OF SUBROUTINE LOGIC    ]

%I0001
--| |---BIT_
        TEST_
        WORD
PRD_CDE-IN  Q-
        LEN
        00001
PICKBIT-BIT

                                     OFFLINE
C:\LESSON                            PRG: LESSON  SUB: SHIP_IT  SIZE:   123  RUNG 0003
REPLACE                                  :            ::
```

14. Output Q is energized if the bit tested is a 1. Press **Relay (Shift-F1)** to display the relay functions. Then, type the reference **%T1** on the command line, and press **F5** to enter a coil at the end of the rung.



```
|RELAY  |TMRCTR |MATH   |RELATN |BITOP  |DATAMV |TABLES |CONVRT |CONTRL |OPN SP
1-] [-  2-]/[-  3       4       5-( )-  6-(SM)- 7-(RM)- 8vert | 9horz - 10more
>

[  START LD SUBROUTINE  SHIP_IT  ]

[     VARIABLE DECLARATIONS      ]

[   START OF SUBROUTINE LOGIC    ]

%I0001
--| |---BIT_
        TEST_
        WORD                                                       %T0001
PRD_CDE-IN  Q-------------------------------------------------------( )-
        LEN
        00001
PICKBIT-BIT

                                     OFFLINE
C:\LESSON                            PRG: LESSON  SUB: SHIP_IT  SIZE:   123  RUNG 0003
REPLACE                       %T0001    :            ::
```

15. The LEN parameter represents the number of words in the string to be tested. This is set at 1 and does not need to be changed for this lesson. However, if you wanted to change the number of words to 5, for example, you would position the cursor on the function, enter **5** on the command line, and press the **Enter** key.

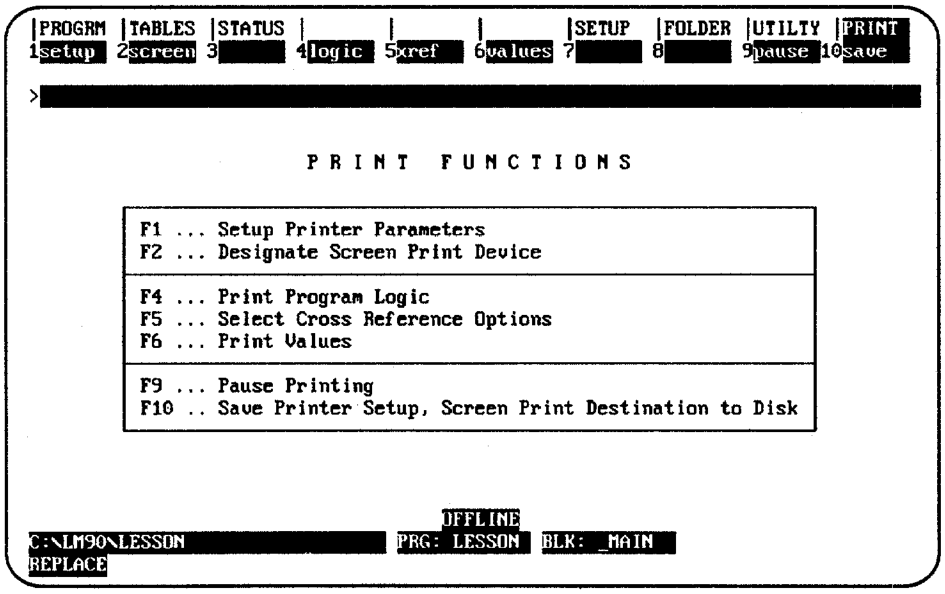
16. That completes the example subroutine. In this subroutine, whenever input %I0001 is set, the bit at the location contained in reference PICKBIT is tested. The bit is part of string PRD_CDE. If it is 1, output Q passes power flow and coil %T0001 is turned on.

17. Press the **Escape** key a total of four times to return to the main menu. If you want to print this program now, continue below.

Printing the Program

If you have a printer connected to the computer, and the Logicmaster 90-30/20/Micro software has been set up to communicate with it, you can print the lesson program now. The software assumes a default printer setup. If that is not correct for your printer, you can enter changes on the Printer Parameters screen, as explained in chapter 9. If the printer has not already been used to print Logicmaster 90 programs, you should check chapter 9, "Print Functions," before continuing.

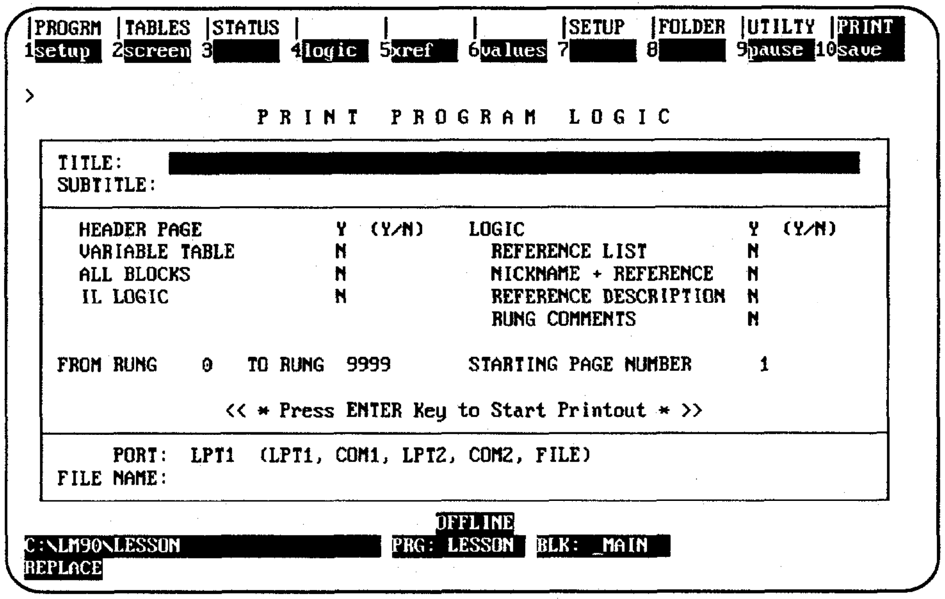
1. To display the Print Function menu, press **Print** (**Shift-F10**).



```
|PROGRM |TABLES |STATUS |      |      |      |SETUP  |FOLDER |UTILTY |PRINT
1setup  2screen 3       4logic 5xref  6values 7      8       9pause 10save
>

                      P R I N T   F U N C T I O N S

         F1 ... Setup Printer Parameters
         F2 ... Designate Screen Print Device

         F4 ... Print Program Logic
         F5 ... Select Cross Reference Options
         F6 ... Print Values

         F9 ... Pause Printing
         F10 .. Save Printer Setup, Screen Print Destination to Disk

                                            OFFLINE
C:\LM90\LESSON                         PRG: LESSON  BLK: _MAIN
REPLACE
```

This menu is used to set printer parameters (F1 and F2), request printing (F4, F5, and F6), pause printing (F9), and save the printer setup (F10). Default printer setup values are supplied with the software, but are easily changed if needed. Chapter 9 explains printer setup in detail.

2. Press **Logic (F4)** to print program logic. Define the printout content on the screen that appears.



```
|PROGRM |TABLES |STATUS |       |       |       |SETUP  |FOLDER |UTILTY |PRINT
1setup  2screen 3       4logic  5xref   6values 7       8       9pause  10save

>
                         P R I N T   P R O G R A M   L O G I C

 TITLE:
 SUBTITLE:

   HEADER PAGE            Y  (Y/N)   LOGIC                      Y  (Y/N)
   VARIABLE TABLE         N            REFERENCE LIST           N
   ALL BLOCKS             N            NICKNAME + REFERENCE     N
   IL LOGIC               N            REFERENCE DESCRIPTION    N
                                       RUNG COMMENTS            N

 FROM RUNG    0   TO RUNG  9999      STARTING PAGE NUMBER      1

                 << * Press ENTER Key to Start Printout * >>

      PORT:  LPT1  (LPT1, COM1, LPT2, COM2, FILE)
 FILE NAME:

                                               OFFLINE
C:\LM90\LESSON                                PRG: LESSON   BLK: _MAIN
REPLACE
```

3. Enter a title for the printout, such as: **PRACTICE PROGRAM.**
4. Both the title and subtitle entries are optional. Skip the subtitle entry.
5. The illustration above shows the default selections for the printout options. To include the example rung explanation in the printout, move the cursor to: "RUNG COMMENTS N". Use the Tab key to toggle the selection from **N** (No) to **Y** (Yes).
6. The other entries shown are suitable for this lesson, so they can be skipped.
7. The listing destination (lower part of the screen) allows you to direct the printout to a serial or parallel port, or to a file. If the destination shown is not correct, use the Down cursor key to move the cursor to *PORT.* Enter the correct port designation.
8. When you are ready to print the program, press the **Enter** key. While the program is being printed, the software displays the message:

```
Listing in progress ... (Listing CAN be paused or aborted).
```

Ending the Lesson

The program you created was saved when you exited the display/edit program function.

Appendix B Configuration Lesson for the Series 90-30 PLC

This appendix contains a tutorial for the configuration software package. The lesson includes:

- Creating a program folder.
- Configuring a CPU module.
- Configuring a 90-30 I/O module.
- Configuring a Genius Communications Module.

You can exit and save the lesson at any time, as explained below. It is not necessary to complete all of the steps.

- To exit and save the lesson, press the **Escape** key.
- To exit without saving the lesson, press **CTRL-Break**.

Help Screens

Logicmaster 90-30/20/Micro software includes detailed Help screens. These Help screens are loaded onto the hard disk of your programmer during the software installation procedure and are readily accessible. To access the Help screens, press **ALT-H** for help, **ALT-I** for instruction mnemonic help, or **ALT-K** for key help.

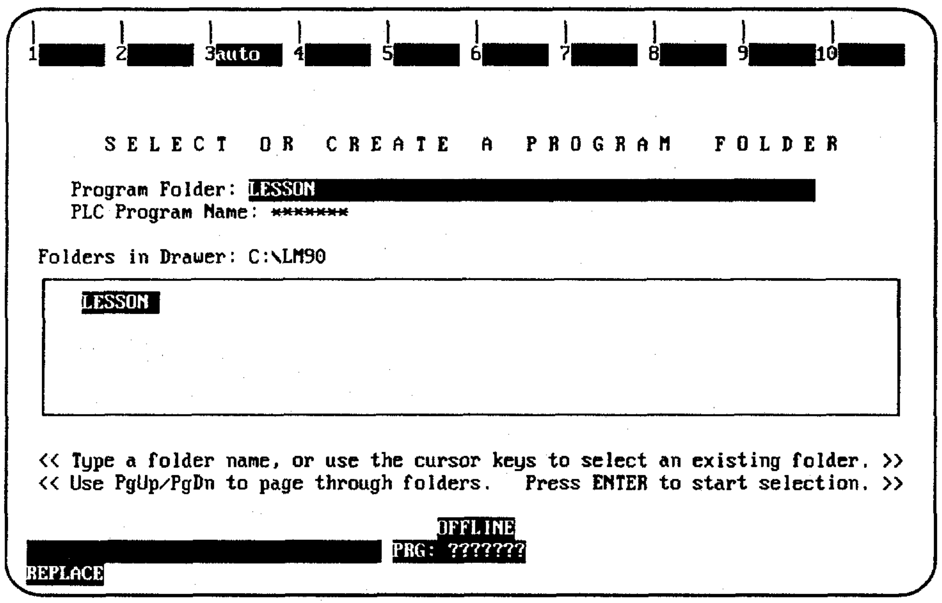
Starting the Lesson

Before you can start, the configuration software must be installed and started up. If that has not been done yet, please turn back to chapter 2 , "Operation," for instructions.

Creating a Program Folder

Each program and its associated configuration are stored as many files. These files are stored in a subdirectory called a program folder. For this lesson, you will use a program folder named **LESSON**.

When Logicmaster 90-30/20/Micro software is started (if the current default directory is an existing program folder), a screen similar to the one shown below appears. If the software is already running, you can display a similar screen from the Program Folder menu by pressing **Folder** (**F8** or **Shift-F8**) from the Configuration Software main menu and then **Select** (**F1**).

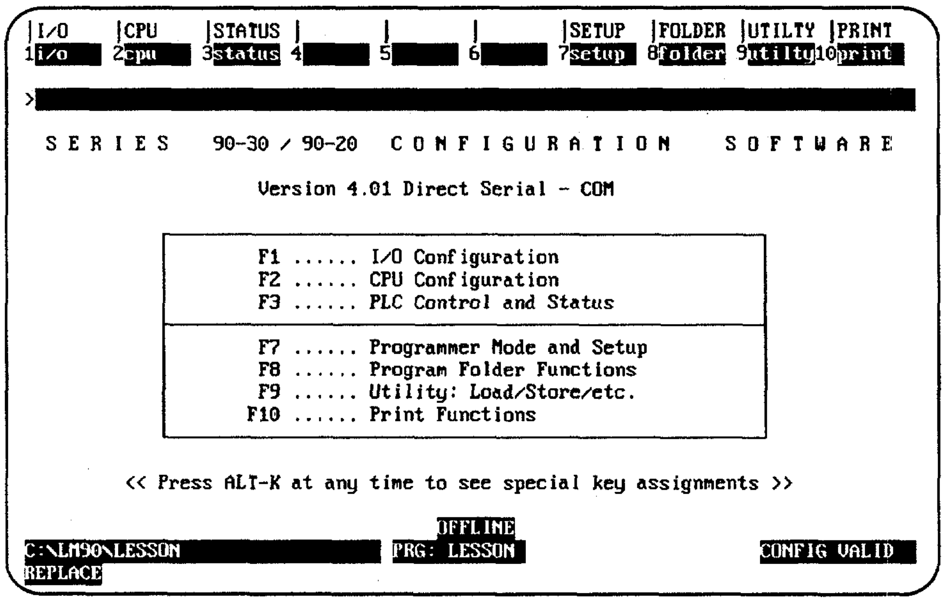


```
1     2     3auto  4     5     6     7     8     9     10

        S E L E C T   O R   C R E A T E   A   P R O G R A M   F O L D E R

   Program Folder: LESSON
   PLC Program Name: ********

Folders in Drawer: C:\LM90

   LESSON

<< Type a folder name, or use the cursor keys to select an existing folder. >>
<< Use PgUp/PgDn to page through folders.   Press ENTER to start selection. >>

                                          OFFLINE
                                       PRG: ???????
REPLACE
```

1. If the current default directory is not an existing program folder, type in a name of seven characters or less for the folder. For this lesson, type **LESSON**.
2. Press the **Enter** key. The following prompt will appear at the top of the screen:

```
Program folder does not exist; create new folder?  (Y/N)
```

3. To create the program folder, enter **Y** (Yes). The software will accept the name and display the Configuration Software main menu.

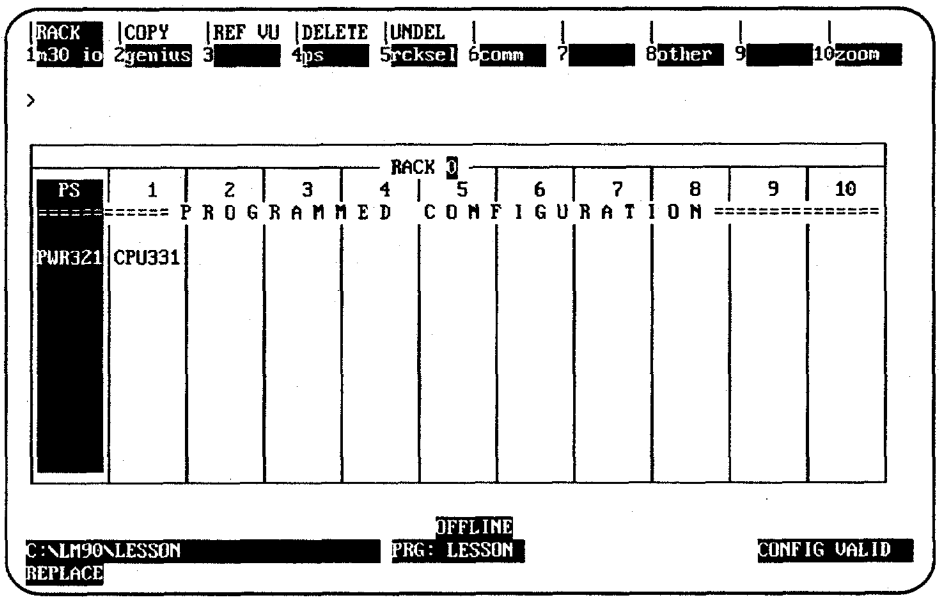


```
|I/O     |CPU     |STATUS |        |        |        |SETUP   |FOLDER  |UTILTY  |PRINT
1i/o     2cpu     3status 4        5        6        7setup   8folder  9utilty  10print
>

 S E R I E S    90-30 / 90-20    C O N F I G U R A T I O N    S O F T W A R E

                     Version 4.01 Direct Serial - COM

                    F1 ...... I/O Configuration
                    F2 ...... CPU Configuration
                    F3 ...... PLC Control and Status

                    F7 ...... Programmer Mode and Setup
                    F8 ...... Program Folder Functions
                    F9 ...... Utility: Load/Store/etc.
                   F10 ...... Print Functions

       << Press ALT-K at any time to see special key assignments >>

                                       OFFLINE
C:\LM90\LESSON                     PRG: LESSON                        CONFIG VALID
REPLACE
```

The name of your new program folder (**LESSON**) should be displayed at the bottom of the screen.

Displaying the Rack Configuration

From the main menu, press I/O (**`F1`** or **`Shift-F1`** from the Folder screen) to display the beginning screen.



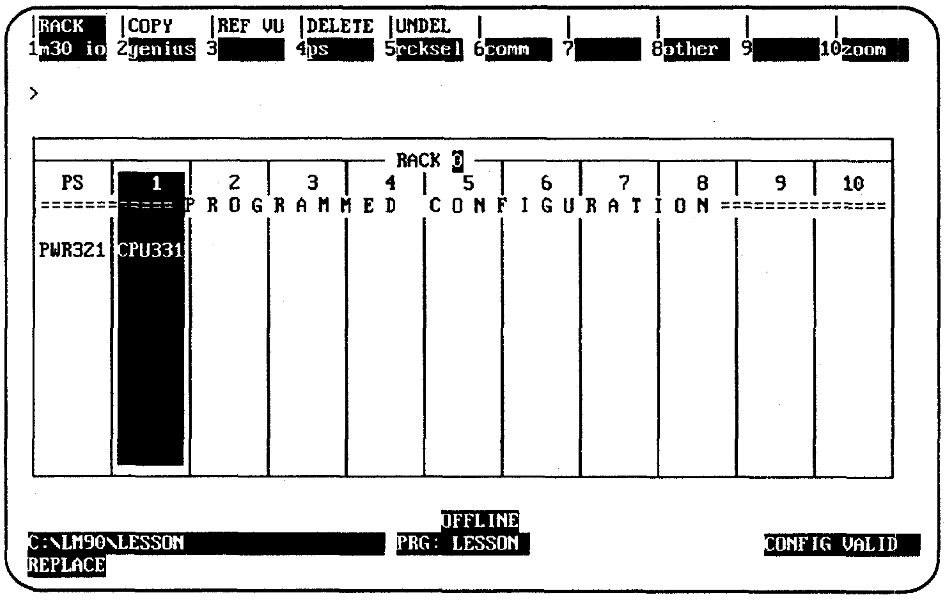
```
|RACK    |COPY    |REF VU  |DELETE  |UNDEL   |        |        |        |        |
1m30 io 2genius  3        4ps      5rcksel  6comm    7        8other   9        10zoom

>

 ------------------------------------ RACK 0 -------------------------------------
  PS   |   1   |   2   |   3   |   4   |   5   |   6   |   7   |   8   |   9   |  10
============ P R O G R A M M E D   C O N F I G U R A T I O N ==============

PWR321 | CPU331|

                                              OFFLINE
C:\LM90\LESSON                               PRG: LESSON                 CONFIG VALID
REPLACE
```

This screen shows the slots available in rack 0, which is the first rack in the system. Use the Page Up or Page Down key, or the Up or Down cursor key, to display the configuration of another rack.

Configuring a CPU Module

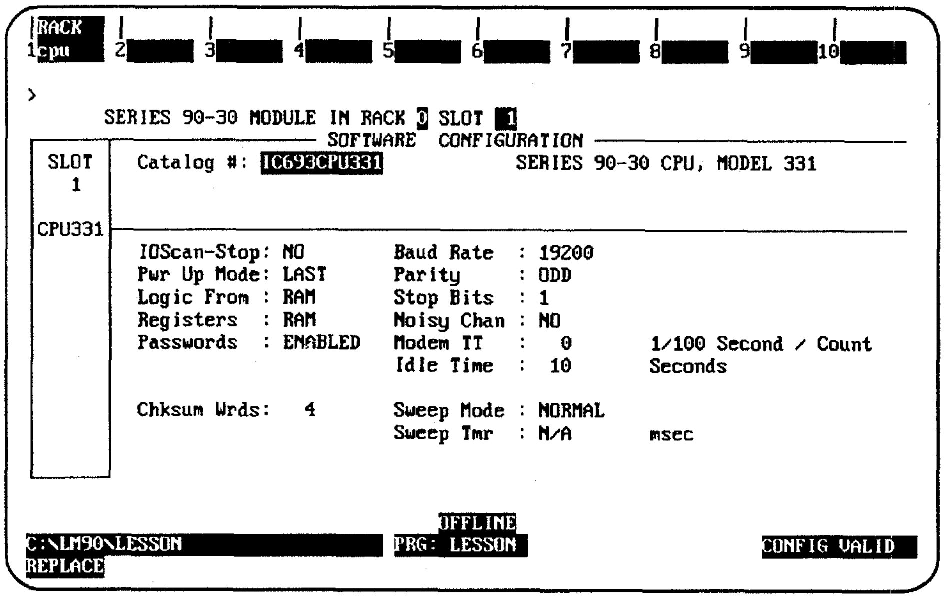
The left slot contains the power supply. Although it requires no additional configuration, its configuration may be changed. Slot 1 contains the CPU module, for which you will enter an example configuration.

1. Press the **Right** cursor movement key once to highlight slot 1.



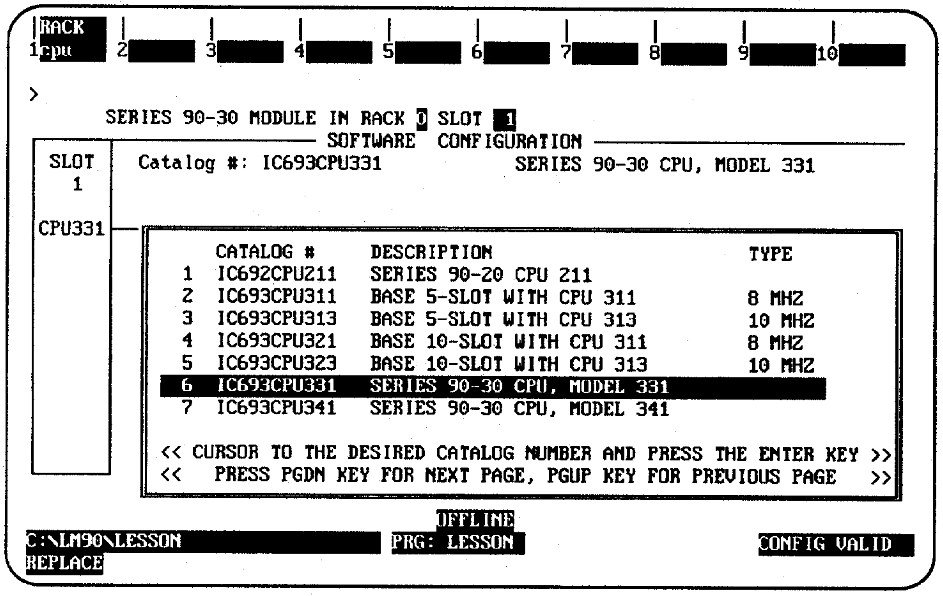
```
|RACK    |COPY    |REF VU  |DELETE  |UNDEL   |        |        |        |        |
1m30 io 2genius  3        4ps      5rcksel  6comm    7        8other   9        10zoom
>
                                  RACK 0
   PS   |   1   |   2   |   3   |   4   |   5   |   6   |   7   |   8   |   9   |  10
=============== P R O G R A M M E D   C O N F I G U R A T I O N ==============
PWR321   CPU331

                                      OFFLINE
C:\LM90\LESSON                        PRG: LESSON                   CONFIG VALID
REPLACE
```

2. Press **Zoom** (`F10`) to display this detail screen.



```
|RACK   |      |      |      |      |      |      |      |      |
1cpu   2      3      4      5      6      7      8      9      10
>
       SERIES 90-30 MODULE IN RACK 0 SLOT 1
                        SOFTWARE  CONFIGURATION
 SLOT    Catalog #: IC693CPU331           SERIES 90-30 CPU, MODEL 331
   1

CPU331
         IOScan-Stop: NO         Baud Rate  : 19200
         Pwr Up Mode: LAST       Parity     : ODD
         Logic From : RAM        Stop Bits  : 1
         Registers  : RAM        Noisy Chan : NO
         Passwords  : ENABLED    Modem TT   :   0       1/100 Second / Count
                                 Idle Time  :  10       Seconds

         Chksum Wrds:    4       Sweep Mode : NORMAL
                                 Sweep Tmr  : N/A       msec

                                      OFFLINE
C:\LM90\LESSON                        PRG: LESSON                   CONFIG VALID
REPLACE
```

3. When each of the parameters on the screen has been entered, press **CPU** **(F1)** to display the list of available CPU modules and their catalog numbers.



```
RACK
1 cpu   2   3   4   5   6   7   8   9   10
>
     SERIES 90-30 MODULE IN RACK 0 SLOT 1
                    SOFTWARE  CONFIGURATION
SLOT    Catalog #: IC693CPU331          SERIES 90-30 CPU, MODEL 331
  1

CPU331
          CATALOG #      DESCRIPTION                      TYPE
       1  IC692CPU211    SERIES 90-20 CPU 211
       2  IC693CPU311    BASE 5-SLOT WITH CPU 311         8 MHZ
       3  IC693CPU313    BASE 5-SLOT WITH CPU 313         10 MHZ
       4  IC693CPU321    BASE 10-SLOT WITH CPU 311        8 MHZ
       5  IC693CPU323    BASE 10-SLOT WITH CPU 313        10 MHZ
       6  IC693CPU331    SERIES 90-30 CPU, MODEL 331
       7  IC693CPU341    SERIES 90-30 CPU, MODEL 341

     << CURSOR TO THE DESIRED CATALOG NUMBER AND PRESS THE ENTER KEY >>
     <<   PRESS PGDN KEY FOR NEXT PAGE, PGUP KEY FOR PREVIOUS PAGE   >>
                                    OFFLINE
C:\LM90\LESSON                      PRG: LESSON              CONFIG VALID
REPLACE
```

4. If you are configuring another type of CPU module, move the cursor to the desired module and press the **Enter** key. Then, enter **Y** (Yes) after the prompt:

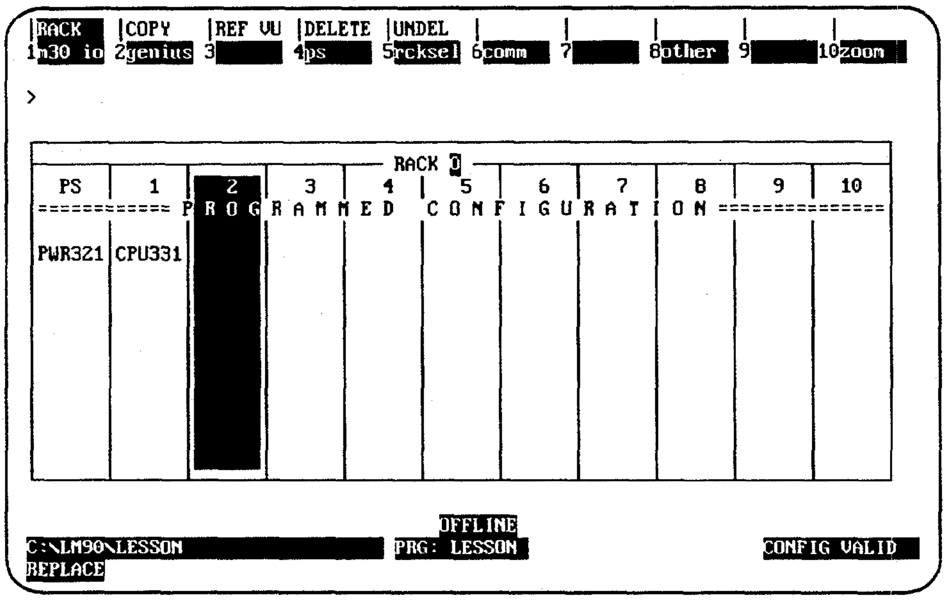
REPLACE displayed module? (Y/N)

5. Press **Rack** **(Shift-F1)** or the **Escape** key to exit from the detail screen and return to the rack display.

Configuring a 90-30 I/O Module

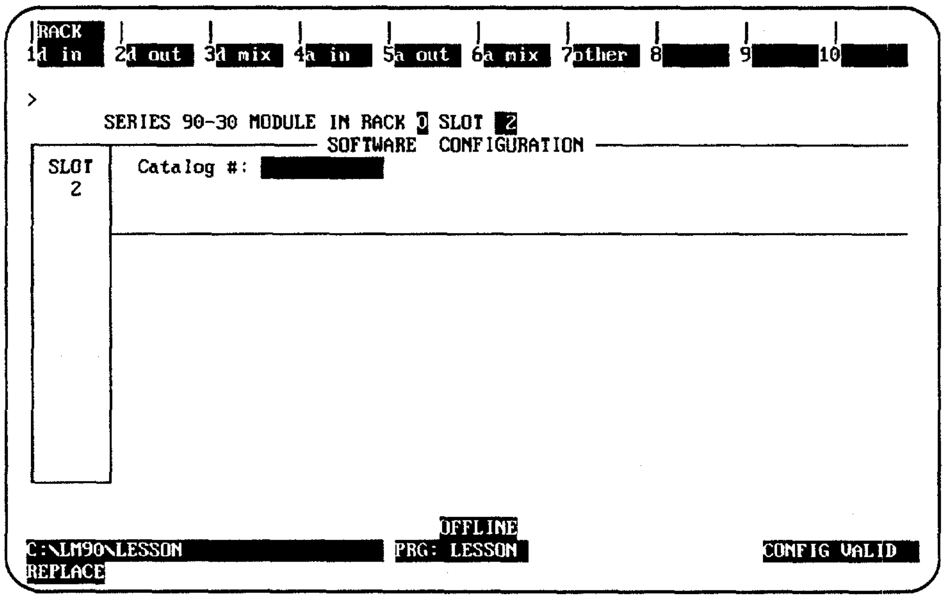
Next, you will configure slot 2 of the main rack for a 90-30 I/O module.

1. Use the Right cursor key to highlight slot 2.



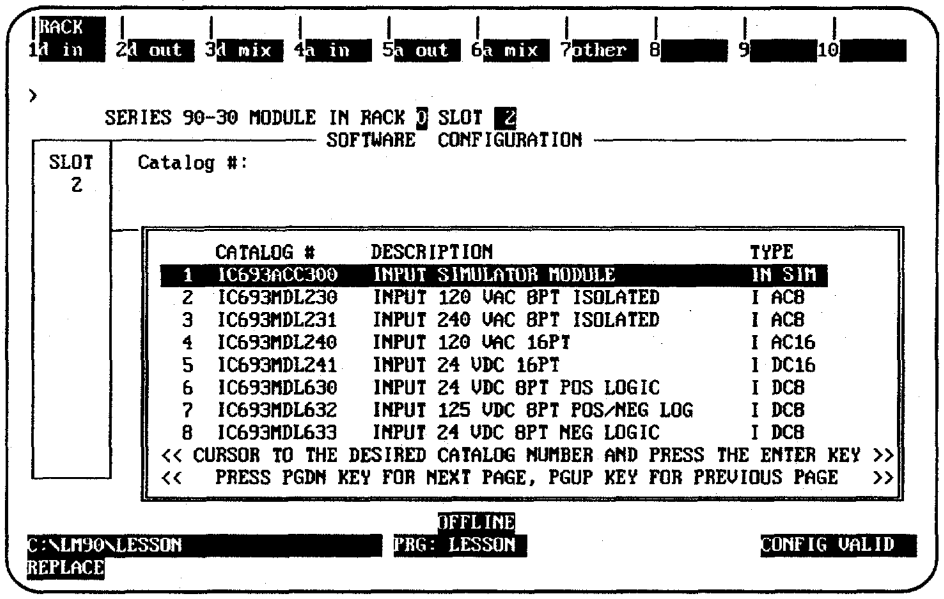
```
|RACK    |COPY    |REF VU  |DELETE  |UNDEL   |        |        |        |        |
1m30 io 2genius  3        4ps      5rcksel  6comm    7        8other   9       10zoom
>
                                          RACK 0
  PS    |   1    |   2    |   3    |   4  |  5     |   6    |   7    |   8    |   9    |  10
=========== P R O G R A M M E D   C O N F I G U R A T I O N ==============
PWR321  |CPU331  |

                                                OFFLINE
C:\LM90\LESSON                                PRG: LESSON                     CONFIG VALID
REPLACE
```

2. Press **Module 30 I/O** `(F1)` to configure a 90-30 I/O Module in slot 2. The software zooms into the detail screen.



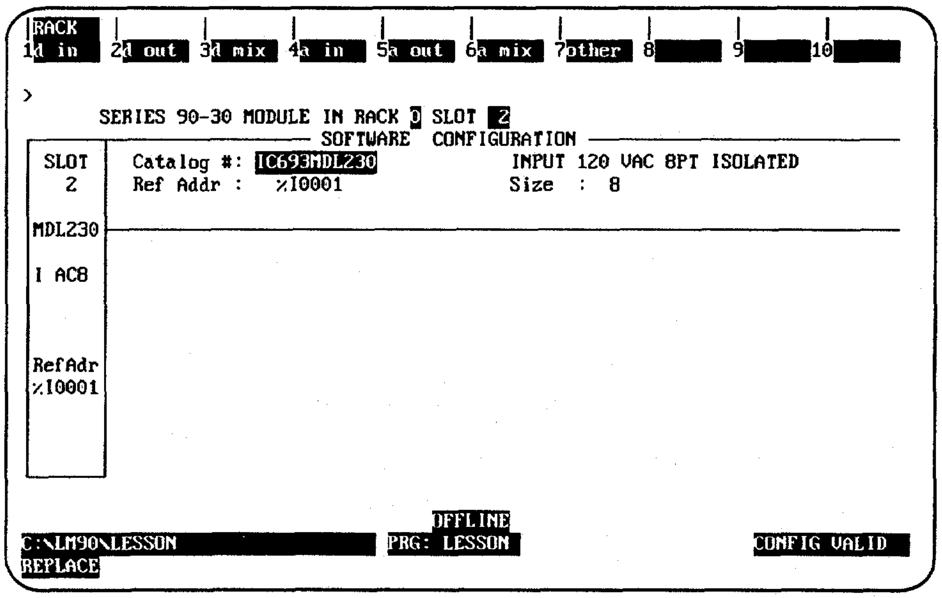
```
|RACK    |        |        |        |        |        |        |        |        |
1d in   2d out   3d mix   4a in    5a out   6a mix   7other   8        9       10
>
         SERIES 90-30 MODULE IN RACK 0 SLOT 2
                              SOFTWARE  CONFIGURATION
  SLOT  |  Catalog #:
    2   |

                                                OFFLINE
C:\LM90\LESSON                                PRG: LESSON                     CONFIG VALID
REPLACE
```

3. The types of modules that might be placed in this slot are shown by the key functions at the top of the screen. For this example, select a Discrete Input module by pressing **F1**. A list of modules is displayed.



```
RACK
1d in  2d out  3d mix  4a in  5a out  6a mix  7other  8      9      10

>
        SERIES 90-30 MODULE IN RACK 0 SLOT 2
                        SOFTWARE  CONFIGURATION
 SLOT    Catalog #:
  2

              CATALOG #     DESCRIPTION                    TYPE
           1  IC693ACC300   INPUT SIMULATOR MODULE         IN SIM
           2  IC693MDL230   INPUT 120 VAC 8PT ISOLATED     I AC8
           3  IC693MDL231   INPUT 240 VAC 8PT ISOLATED     I AC8
           4  IC693MDL240   INPUT 120 VAC 16PT             I AC16
           5  IC693MDL241   INPUT 24 VDC 16PT              I DC16
           6  IC693MDL630   INPUT 24 VDC 8PT POS LOGIC     I DC8
           7  IC693MDL632   INPUT 125 VDC 8PT POS/NEG LOG  I DC8
           8  IC693MDL633   INPUT 24 VDC 8PT NEG LOGIC     I DC8
          << CURSOR TO THE DESIRED CATALOG NUMBER AND PRESS THE ENTER KEY >>
          <<  PRESS PGDN KEY FOR NEXT PAGE, PGUP KEY FOR PREVIOUS PAGE   >>
                                 OFFLINE
C:\LM90\LESSON                  PRG: LESSON              CONFIG VALID
REPLACE
```

4. Select a module by moving the cursor to any entry and pressing the **Enter** key. Its catalog number will appear in reverse video in the proper field. For this example, move the cursor to **IC693MDL230** and press the **Enter** key. The following screen is displayed:



```
RACK
1d in  2d out  3d mix  4a in  5a out  6a mix  7other  8      9      10

>
        SERIES 90-30 MODULE IN RACK 0 SLOT 2
                        SOFTWARE  CONFIGURATION
 SLOT    Catalog #: IC693MDL230          INPUT 120 VAC 8PT ISOLATED
  2      Ref Addr :  %I0001              Size  :  8

MDL230

I AC8


RefAdr
%I0001

                                 OFFLINE
C:\LM90\LESSON                  PRG: LESSON              CONFIG VALID
REPLACE
```

5. Assign a beginning I/O reference address to this module. Move the cursor to:

```
Ref Addr :  %I00001
```

For this example, you will purposely enter an incorrect I/O reference. First, move the small cursor to the right in the ***Reference Address*** field by pressing the **CTRL** and Right cursor keys. Enter `1` and then `6` as the last two digits. Now the reference address should look like this:

```
Ref Addr :  %I00016
```

This reference address is incorrect because it does not begin on a byte boundary. (A byte boundary is a number which is one greater than an integer multiple of 8; for example, 1, 9, 17, or 25).

6. Watch the reference address. Notice that when you press the Down cursor key once, the software automatically changes the reference address to the next lowest byte boundary:

```
Ref Addr :  %I00009
```

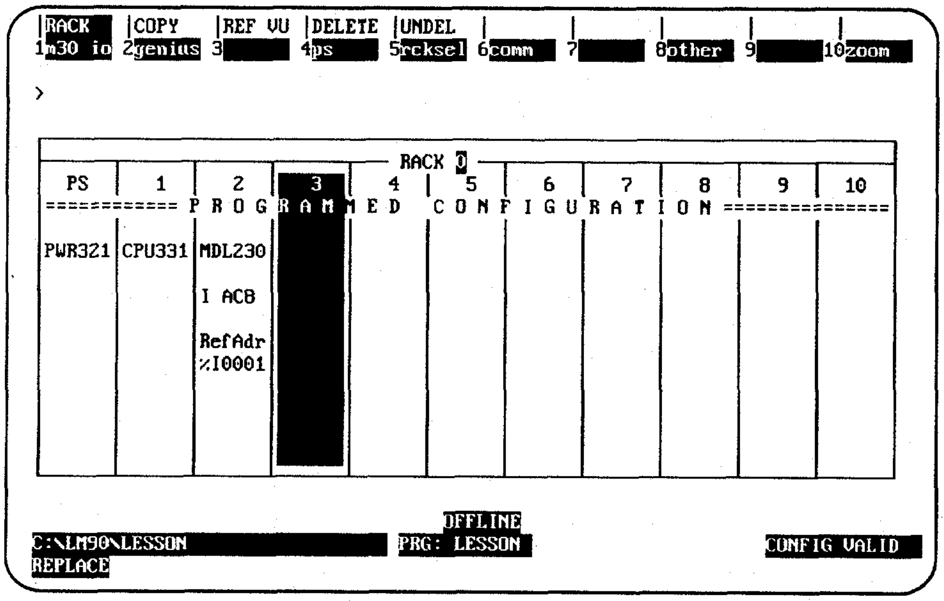
and the message "The Reference Address has been adjusted to be byte aligned" will be displayed.

7. For this lesson, the rest of the entries on this screen need not be changed. Press the **Escape** key to return to the rack display.

Configuring a Genius Communications Module

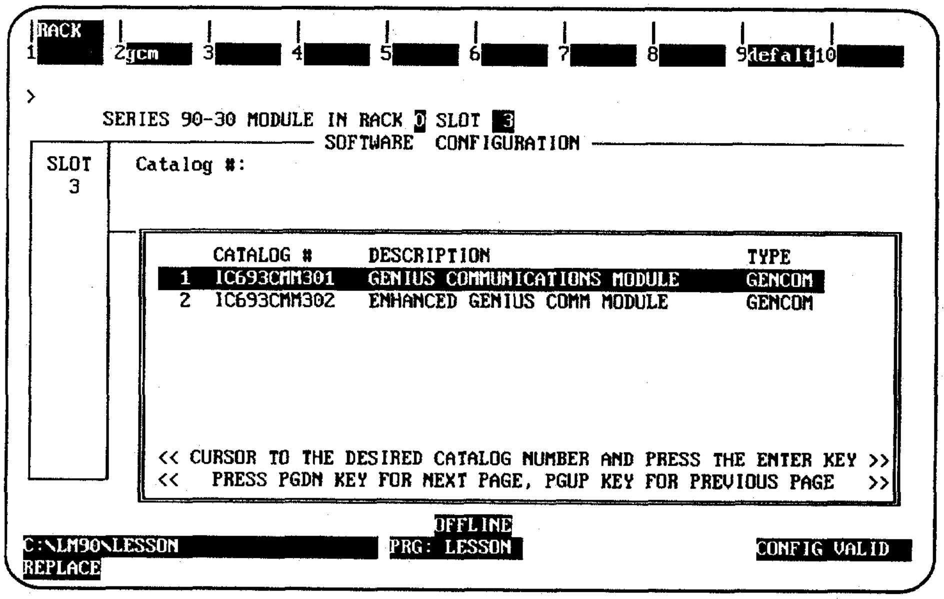
Next, you will configure slot 3 of the main rack for a Genius Communications Module.

1. Use the Right cursor key to highlight slot 3.



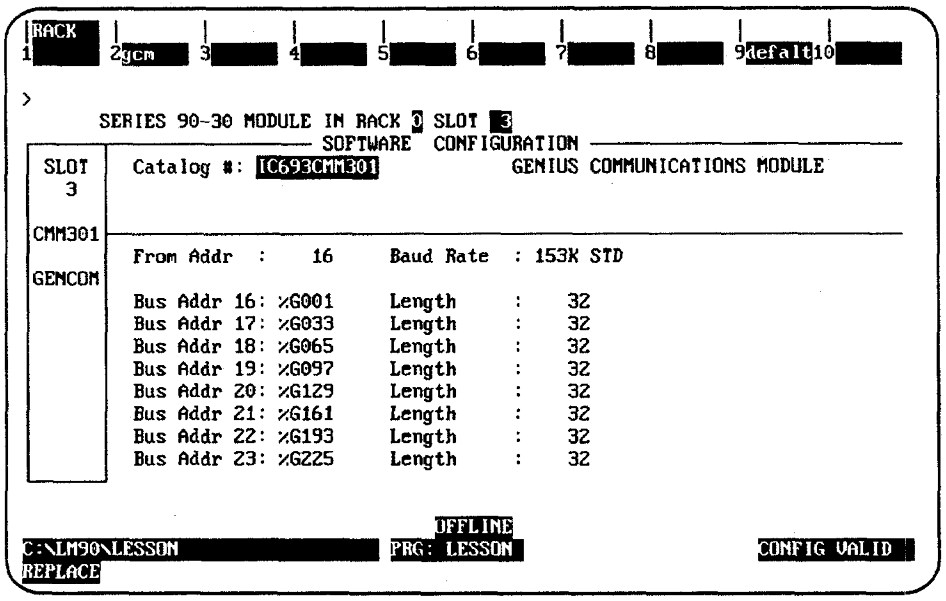
```
|RACK    |COPY    |REF VU  |DELETE  |UNDEL   |        |        |        |        |
1m30 io  2genius  3        4ps      5rcksel  6comm    7        8other   9        10zoom

>

                                    RACK 0
  PS   |   1   |   2   |   3   |   4   |   5   |   6   |   7   |   8   |   9   |   10
=========== P R O G R A M M E D   C O N F I G U R A T I O N ==============
PWR321 |CPU331 |MDL230 |
               |I AC8  |
               |RefAdr |
               |%I0001 |

                                            OFFLINE
C:\LM90\LESSON                              PRG: LESSON                 CONFIG VALID
REPLACE
```

2. Press **Genius (F2)** from the I/O Rack Configuration screen. Then, press **GCM (F2)** to display the catalog number of the GCM.



```
|RACK    |        |        |        |        |        |        |        |        |
1        2gcm     3        4        5        6        7        8        9default 10

>
     SERIES 90-30 MODULE IN RACK 0 SLOT 3
                             SOFTWARE  CONFIGURATION
 SLOT   Catalog #:
   3

             CATALOG #      DESCRIPTION                        TYPE
          1  IC693CMM301    GENIUS COMMUNICATIONS MODULE       GENCOM
          2  IC693CMM302    ENHANCED GENIUS COMM MODULE        GENCOM

       << CURSOR TO THE DESIRED CATALOG NUMBER AND PRESS THE ENTER KEY >>
       <<   PRESS PGDN KEY FOR NEXT PAGE, PGUP KEY FOR PREVIOUS PAGE   >>
                                            OFFLINE
C:\LM90\LESSON                              PRG: LESSON                 CONFIG VALID
REPLACE
```

3. Press the **Enter** key to enter the catalog number shown in reverse video. The following detail screen is displayed.



```
|RACK |     |    |    |    |    |    |    |        |
1     2gcm  3    4    5    6    7    8    9defalt10

>
        SERIES 90-30 MODULE IN RACK 0 SLOT 3
                          SOFTWARE  CONFIGURATION
 SLOT   Catalog #: IC693CMM301             GENIUS COMMUNICATIONS MODULE
  3

CMM301
        From Addr   :    16       Baud Rate   : 153K STD
GENCOM
        Bus Addr 16: %G001        Length      :    32
        Bus Addr 17: %G033        Length      :    32
        Bus Addr 18: %G065        Length      :    32
        Bus Addr 19: %G097        Length      :    32
        Bus Addr 20: %G129        Length      :    32
        Bus Addr 21: %G161        Length      :    32
        Bus Addr 22: %G193        Length      :    32
        Bus Addr 23: %G225        Length      :    32

                                  OFFLINE
C:\LM90\LESSON                    PRG: LESSON              CONFIG VALID
REPLACE
```

4. Complete the detail screen. To do this, you must be familiar with information in the *Genius Communications Module Manual,* GFK-0412. For this example, we will assume the default entries are correct.

5. Press **Rack** **(`Shift-F1`)** or the **Escape** key to return to the rack display.

6. Press the **Escape** key from the rack display (or **CTRL-U**) to save the created configuration.

Ending the Lesson

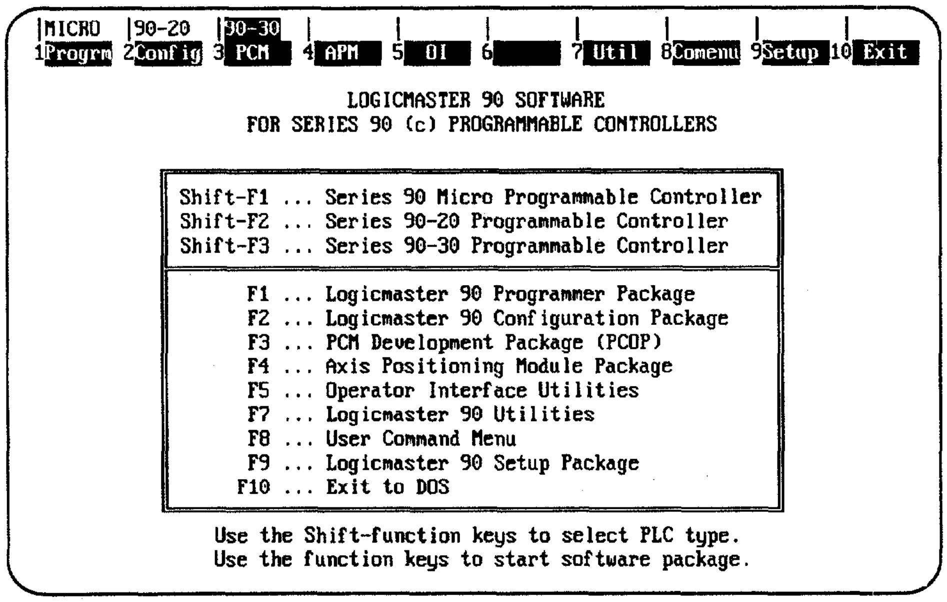
The configuration created was saved when you pressed **Rack (`Shift-F1`)** or the **Escape** key to return to the rack display. This is the end of the lesson.

Appendix C Programmer Environment Setup

A default setup file is created during installation. Only in special circumstances will this file need to be changed. Follow the steps below in order to edit the setup file.

Displaying the Setup Screen

1. After completing the installation and restarting the computer as described, at the MS-DOS prompt, type **LM90** and press the **Enter** key. A menu of Series 90 PLCs and functions is displayed.



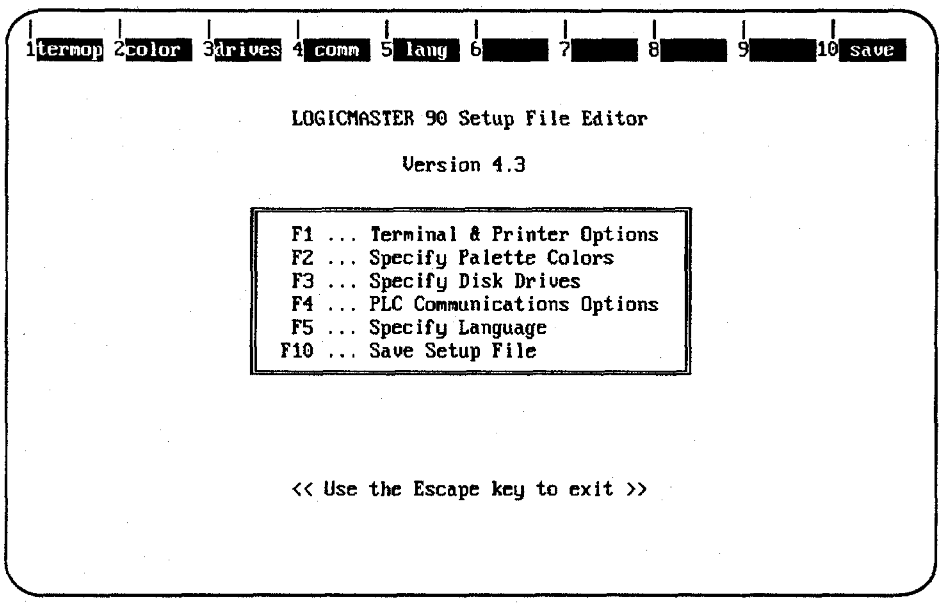
|MICRO |90-20 |90-30 | | | | | | |
1Progrm 2Config 3 PCM 4 APM 5 OI 6 7 Util 8Comenu 9Setup 10 Exit

```
LOGICMASTER 90 SOFTWARE
FOR SERIES 90 (c) PROGRAMMABLE CONTROLLERS

Shift-F1 ... Series 90 Micro Programmable Controller
Shift-F2 ... Series 90-20 Programmable Controller
Shift-F3 ... Series 90-30 Programmable Controller

      F1 ... Logicmaster 90 Programmer Package
      F2 ... Logicmaster 90 Configuration Package
      F3 ... PCM Development Package (PCOP)
      F4 ... Axis Positioning Module Package
      F5 ... Operator Interface Utilities
      F7 ... Logicmaster 90 Utilities
      F8 ... User Command Menu
      F9 ... Logicmaster 90 Setup Package
     F10 ... Exit to DOS

Use the Shift-function keys to select PLC type.
Use the function keys to start software package.
```

Note

Unlike the setup features discussed in chapter 6, you must begin the setup steps detailed in this appendix by entering the "Logicmaster 90 Setup Package" by pressing *F9 from the Logicmaster main menu, not from within the programmer.*

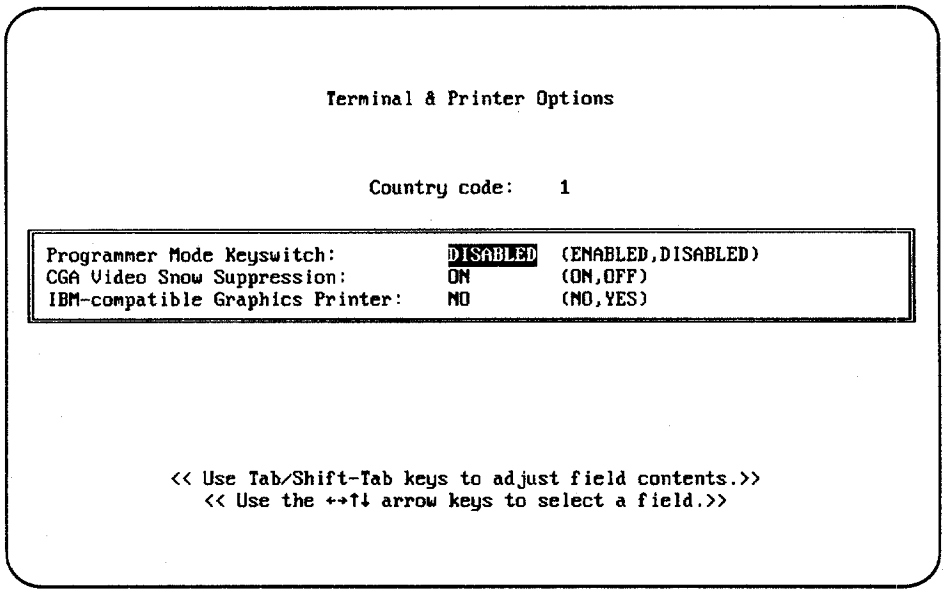
2. Press **Setup** (`F9`) to select the setup function. The following menu is displayed:



```
1termop 2color 3drives 4 comm 5 lang 6      7      8      9      10 save

                    LOGICMASTER 90 Setup File Editor

                              Version 4.3

                     F1 ... Terminal & Printer Options
                     F2 ... Specify Palette Colors
                     F3 ... Specify Disk Drives
                     F4 ... PLC Communications Options
                     F5 ... Specify Language
                    F10 ... Save Setup File

                    << Use the Escape key to exit >>
```

Selecting Terminal and Printer Options

To select terminal and printer options, press **Terminal Options** **(F1)** from the Setup File Editor menu.



```
                    Terminal & Printer Options


                         Country code:     1

 Programmer Mode Keyswitch:          DISABLED  (ENABLED,DISABLED)
 CGA Video Snow Suppression:         ON        (ON,OFF)
 IBM-compatible Graphics Printer:    NO        (NO,YES)




          << Use Tab/Shift-Tab keys to adjust field contents.>>
             << Use the ←→↑↓ arrow keys to select a field.>>
```

Use the cursor keys to move from one option field to another. Selected fields are shown in reverse video on the display screen. Then, use the **Tab** and **Shift-Tab** keys to toggle each selection.

Field	Description
Programmer Mode Keyswitch	Logicmaster 90-30/20/Micro software may be run on machines with or without a keyswitch to specify the operational mode (**ONLINE**, **MONITOR**, or **OFFLINE**). Choices for this field are **ENABLED** or **DISABLED***. For a Workmaster or CIMSTAR I industrial computer with a keyswitch, the keyswitch should be enabled. For computers without a keyswitch, it should be disabled. Note: Enabling the keyswitch disables the use of ALT-M to change modes. You must disable the keyswitch in order to use ALT-M to change operational modes on a computer with a keyswitch.
CGA Video Snow Suppression	This field is important only if your personal computer uses a CGA (Color Graphics Adapter) compatible video adapter. If you have a CGA-compatible adapter that does not need delays to avoid video flicker (referred to as snow), you may increase the performance of Logicmaster 90-30/20/Micro software by using this entry to disable snow control. Choices for this field are **ON*** or **OFF**. For a system using an IBM compatible Color Graphics Adapter, snow control should be enabled by selecting **ON**. For systems which have eliminated this problem, the selection should be **OFF**.
IBM Compatible Graphics Printer	If your printer is capable of producing IBM compatible graphics characters, you may improve the appearance of your ladder logic printouts by enabling this table has no title this option. Choices for this field are **YES** or **NO***.

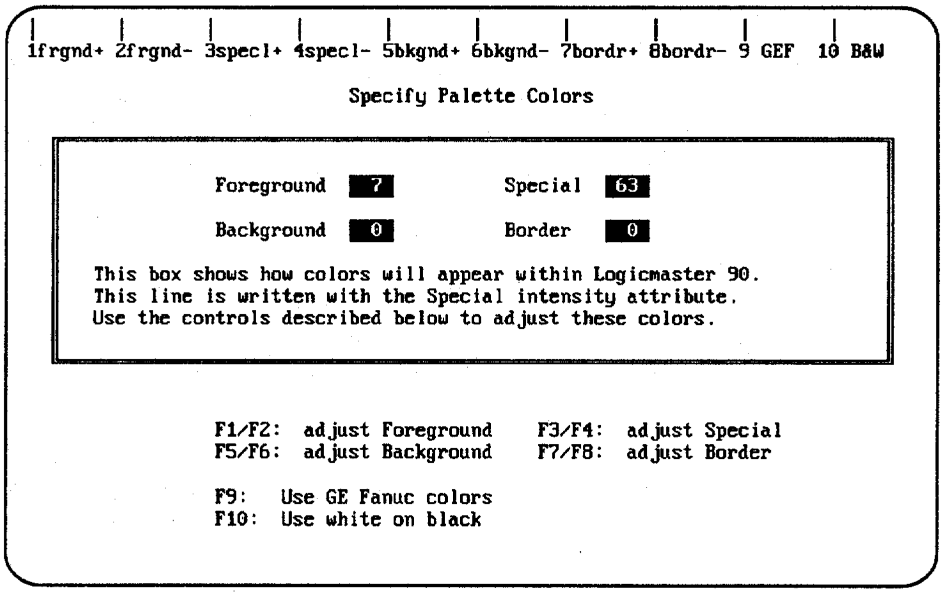
* Default Selection

When the selections are complete, press the **Escape** key to return to the Setup File Editor menu.

Specifying Palette Colors

If your personal computer has a color monitor and the video adapter is an EGA (Enhanced Graphics Adapter) or VGA (Video Graphics Array), you may select the colors that will be used in the Logicmaster 90 displays.

To specify palette colors, press **Colors** **(F2)** from the Setup File Editor menu.



```
1frgnd+ 2frgnd- 3specl+ 4specl- 5bkgnd+ 6bkgnd- 7bordr+ 8bordr- 9 GEF  10 B&W

                          Specify Palette Colors

              Foreground   7              Special   63

              Background   0              Border    0

     This box shows how colors will appear within Logicmaster 90.
     This line is written with the Special intensity attribute.
     Use the controls described below to adjust these colors.

              F1/F2:  adjust Foreground    F3/F4:  adjust Special
              F5/F6:  adjust Background    F7/F8:  adjust Border

              F9:   Use GE Fanuc colors
              F10:  Use white on black
```

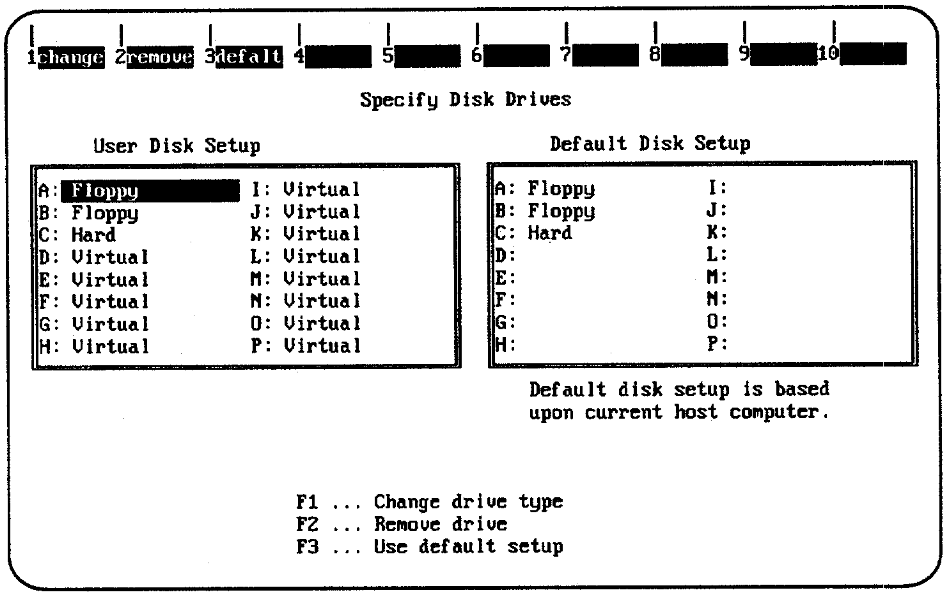
The middle portion of the screen, enclosed in a box, provides a sample display using the currently selected colors. The numeric values stored in the palette registers control the foreground, background, special, and screen border colors.

The color values in the palette registers may be changed by pressing the function keys (**F1** through **F10**), as described in the lower portion of the screen. The function keys are paired; one key increases the value while the other decreases it. The values may be adjusted within the range `0` to `63`. As the values are changed, the colors in the sample display box are updated accordingly.

You may wish to experiment with different color combinations. When you are satisfied with your selections, press the **Escape** key to return to the Setup menu. The color values will be saved along with the other setup information; these color values will be used each time you start up the Logicmaster 90-30/20/Micro software.

Completing the Disk Drive Setup

To complete the disk drive setup, press **Drives (F3)** from the Setup File Editor menu.



```
                         Specify Disk Drives

     User Disk Setup                         Default Disk Setup

A: Floppy          I: Virtual       A: Floppy          I:
B: Floppy          J: Virtual       B: Floppy          J:
C: Hard            K: Virtual       C: Hard            K:
D: Virtual         L: Virtual       D:                 L:
E: Virtual         M: Virtual       E:                 M:
F: Virtual         N: Virtual       F:                 N:
G: Virtual         O: Virtual       G:                 O:
H: Virtual         P: Virtual       H:                 P:

                                       Default disk setup is based
                                       upon current host computer.

                    F1 ... Change drive type
                    F2 ... Remove drive
                    F3 ... Use default setup
```

1. If the drives shown on the left of your screen are correct, press **Default (F3)**. If the current selections are not correct, change them as instructed.
2. Press the **Escape** key to return to the Setup File Editor menu.

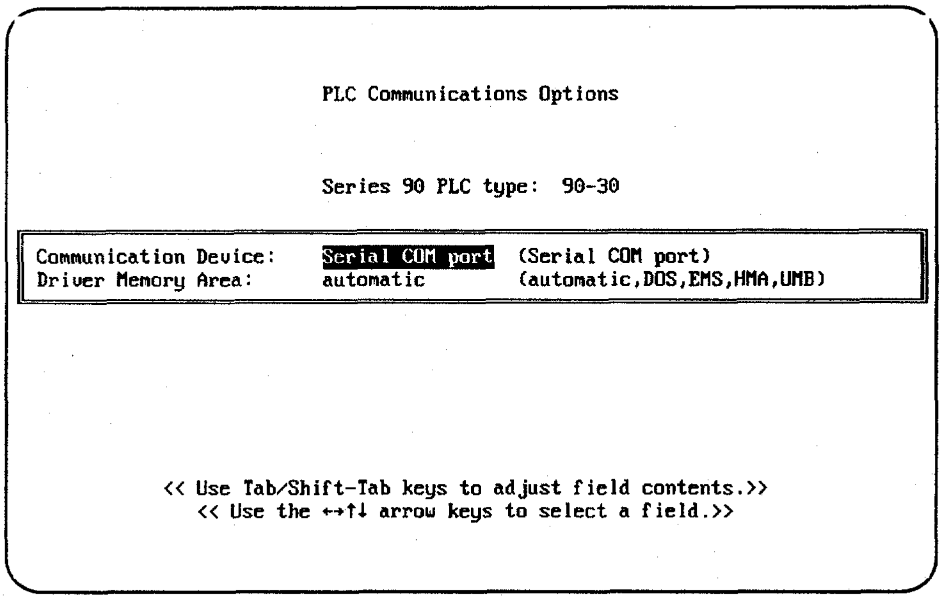
PLC Communications Options

The PLC Communications Options screen enables you to select whether to communicate with the PLC through a WSI Board or serial COM port and to select one of four different interrupt lines on the PC backplane. To display this screen, press **PLC Communications Options** `(F4)` *from the Setup File Editor menu.*

Remember, unlike the serial port setup discussed in chapter 6, to access this screen you must enter **Setup** `(F9)` *from the main menu* which takes you to the Setup File Editor screen, then press **PLC Communications Options** `(F4)` —see pages C-1 – C-2 for more information.

Standard Serial COM Port Installed

In the following example screen, only the standard serial communications version of software is installed.



```
                      PLC Communications Options


                      Series 90 PLC type:  90-30

 Communication Device:  Serial COM port  (Serial COM port)
 Driver Memory Area:    automatic        (automatic,DOS,EMS,HMA,UMB)




            << Use Tab/Shift-Tab keys to adjust field contents.>>
               << Use the ↔↑↓ arrow keys to select a field.>>
```

Field	Description
Communication Device	When both the WSI and standard serial communications versions of Logicmaster 90-30/20/Micro software are installed, you must choose whether to communicate with the PLC through a **`WSI Board*`** or **`serial COM port`**.
Driver Memory Area	This field shows the current selection for which area of memory the communications driver will load into. The default and recommended setting is "**automatic**" as shown above.

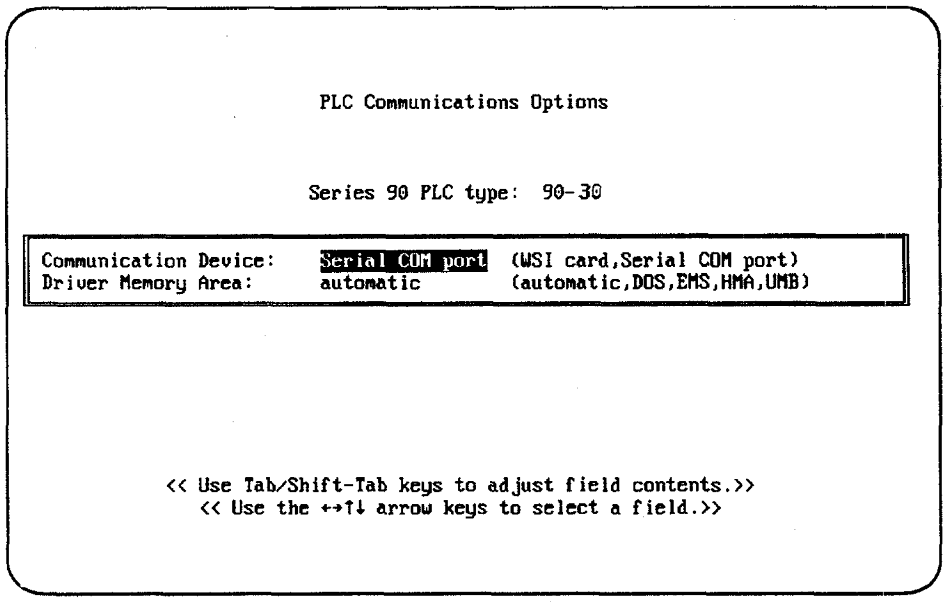
* Default Selection

Standard Serial COM Port and Driver Memory Area

To change the serial COM memory area, use the **Down Arrow** key to move the cursor to the "Driver Memory Area" field. (Selected fields are shown in reverse video on the display screen.) Then, use the **Tab** and **Shift-Tab** keys to toggle each selection. As noted above, the "automatic" setting is recommended; this lets Logicmaster determine which area of memory it is best to place the driver. For information about the different memory configuration options, read pages 6-6 and following in this manual.

PLC Communications Options—
Both WSI and Serial COM Port Installed

In the following example, both the WSI and Standard Serial Communications versions of Logicmaster 90-30/20/Micro software are installed, and the serial COM port was chosen as the communication driver.



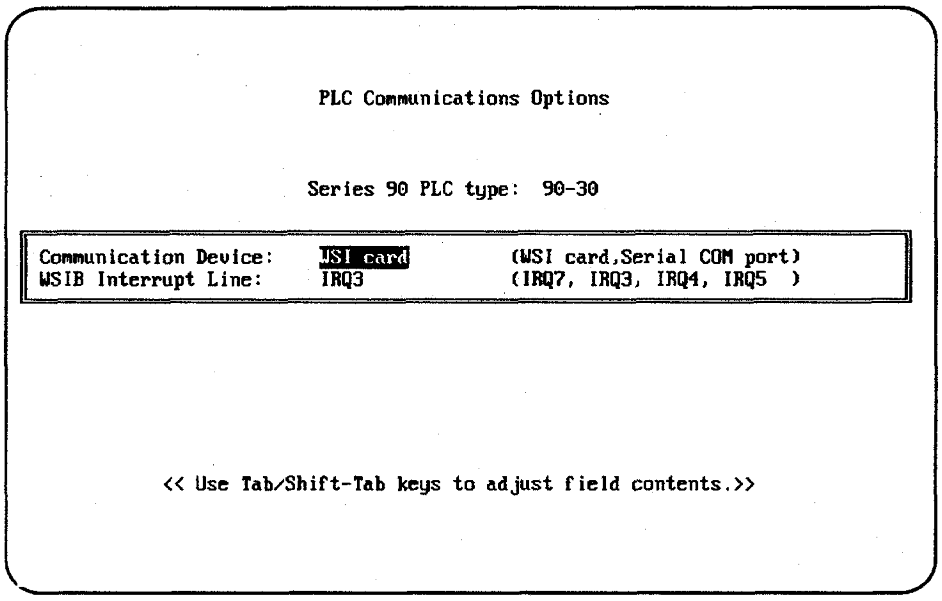
```
                    PLC Communications Options


                   Series 90 PLC type:  90-30

 Communication Device:    Serial COM port  (WSI card,Serial COM port)
 Driver Memory Area:      automatic        (automatic,DOS,EMS,HMA,UMB)




           << Use Tab/Shift-Tab keys to adjust field contents.>>
              << Use the ←→↑↓ arrow keys to select a field.>>
```

Note

To change the selection of "Communication Device," use the **Tab** and **Shift-Tab** keys to toggle the selection between "Serial COM port" and "WSI." This, of course, is only applicable *if* you have both the WSI and the Standard Serial COM Port installed.

PLC Communications— WSI Version with Configurable Interrupt Request

In the next example, the WSI version of software is installed and selected, and a WSIB2 Board was used. Note that the default value **IRQ3** appears in the "WSIB Interrupt Line" field. This value can be changed by using the **Tab** and **Shift-Tab** keys to toggle each selection.



```
                    PLC Communications Options

                    Series 90 PLC type:  90-30

 Communication Device:    WSI card       (WSI card,Serial COM port)
 WSIB Interrupt Line:     IRQ3           (IRQ7, IRQ3, IRQ4, IRQ5  )

          << Use Tab/Shift-Tab keys to adjust field contents.>>
```

Field	Description
Communication Device	When both the WSI and standard serial communications versions of Logicmaster 90-30/20/Micro software are installed, you must choose whether to communicate with the PLC through a **WSI Board*** or **serial COM port**.
WSIB Interrupt Line	In a PC or PC-AT computer, the WSI Board can be configured to use four different interrupt lines on the PC backplane. Choices are **IRQ3**, **IRQ4**, **IRQ5**, or **IRQ7***. This option is available only if you are using the WSIB2 Board for communications. If you have the older WSIB1 Board, the interrupt line is displayed as IRQ3 and is not changeable. If you have selected a serial COM port for communications, this option is not displayed.

* Default Selection

Note

When the WSI Version of Release 4 or later of Logicmaster 90-30/20/Micro software is used on a Workmaster II industrial computer or other microchannel personal computer with a WS9A2 board, the selection of an interrupt line on the PLC Communications Options screen must be set to IRQ3. Failure to select IRQ3 will result in improper setup of the board, possible lockup of the Logicmaster software during startup, and failure to communicate with the PLC.

FANUC Logicmaster™ Key Help (ALT-K) GFJ-055B

Key Sequence	Description
Keys Available throughout the Software Package	
ALT-A	Abort.
ALT-C	Clear field.
ALT-M	Change Programmer mode.
ALT-R	Change PLC Run/Stop state.
ALT-E	Toggle status area.
ALT-J	Toggle the command line.
ALT-L	List directory files.
ALT-P	Print screen.
ALT-H	Help.
ALT-K	Key help.
ALT-I	Instruction mnemonic help.
ALT-T	Start Teach mode.
ALT-Q	Stop Teach mode.
ALT-n	Playback file n (n = 0 thru 9).
CTRL-Break	Exit package.
Esc	Zoom out.
CTRL-Home	Previous command line.
CTRL-End	Next command line.
CTRL-←	Cursor left within the field.
CTRL-→	Cursor right within the field.
CTRL-D	Decrement reference address.
CTRL-U	Increment reference address.
Tab	Change/increment field contents.
Shift-Tab	Change/decrement field contents.
Enter	Accept field contents.
CTRL-E	Display last system error.
F12 or Keypad −	Toggle discrete reference.
F11 or Keypad *	Override discrete reference.
Keys Available in the Program Editor Only	
ALT-B	Toggle text editor bell.
ALT-D	Delete rung element.
ALT-S	Store block to PLC and disk.
ALT-X	Display zoom level.
ALT-N	Toggle nickname/reference address.
ALT-U	Update disk.
ALT-V	Variable table window.
Keypad +	Accept rung.
Enter	Accept rung.
CTRL-PgUp	Previous rung.
CTRL-PgDn	Next rung.
	Horizontal shunt.
|	Vertical shunt.
Tab	Go to the next operand field.
Special Keys	
ALT-O	Password override. Available only on the Password screen in the configuration software.

FANUC Logicmaster™ Key Help (ALT-K) GFJ-055B

Key Sequence | Description
Keys Available throughout the Software Package
ALT-A Abort. ALT-C Clear field. ALT-M Change Programmer mode. ALT-R Change PLC Run/Stop state. ALT-E Toggle status area. ALT-J Toggle the command line. ALT-L List directory files. ALT-P Print screen. ALT-H Help. ALT-K Key help. ALT-I Instruction mnemonic help. ALT-T Start Teach mode. ALT-Q Stop Teach mode. ALT-n Playback file n (n = 0 thru 9). CTRL-Break Exit package. Esc Zoom out. CTRL-Home Previous command line. CTRL-End Next command line. CTRL-← Cursor left within the field. CTRL-→ Cursor right within the field. CTRL-D Decrement reference address. CTRL-U Increment reference address. Tab Change/increment field contents. Shift-Tab Change/decrement field contents. Enter Accept field contents. CTRL-E Display last system error. F12 or Keypad − Toggle discrete reference. F11 or Keypad * Override discrete reference.
Keys Available in the Program Editor Only
ALT-B Toggle text editor bell. ALT-D Delete rung element. ALT-S Store block to PLC and disk. ALT-X Display zoom level. ALT-N Toggle nickname/reference address. ALT-U Update disk. ALT-V Variable table window. Keypad + Accept rung. Enter Accept rung. CTRL-PgUp Previous rung. CTRL-PgDn Next rung. Horizontal shunt. | Vertical shunt. Tab Go to the next operand field.
Special Keys
ALT-O Password override. Available only on the Password screen in the configuration software.

FANUC Logicmaster™ Key Help (ALT-K) GFJ-055B

Key Sequence | Description
Keys Available throughout the Software Package
ALT-A Abort. ALT-C Clear field. ALT-M Change Programmer mode. ALT-R Change PLC Run/Stop state. ALT-E Toggle status area. ALT-J Toggle the command line. ALT-L List directory files. ALT-P Print screen. ALT-H Help. ALT-K Key help. ALT-I Instruction mnemonic help. ALT-T Start Teach mode. ALT-Q Stop Teach mode. ALT-n Playback file n (n = 0 thru 9). CTRL-Break Exit package. Esc Zoom out. CTRL-Home Previous command line. CTRL-End Next command line. CTRL-← Cursor left within the field. CTRL-→ Cursor right within the field. CTRL-D Decrement reference address. CTRL-U Increment reference address. Tab Change/increment field contents. Shift-Tab Change/decrement field contents. Enter Accept field contents. CTRL-E Display last system error. F12 or Keypad − Toggle discrete reference. F11 or Keypad * Override discrete reference.
Keys Available in the Program Editor Only
ALT-B Toggle text editor bell. ALT-D Delete rung element. ALT-S Store block to PLC and disk. ALT-X Display zoom level. ALT-N Toggle nickname/reference address. ALT-U Update disk. ALT-V Variable table window. Keypad + Accept rung. Enter Accept rung. CTRL-PgUp Previous rung. CTRL-PgDn Next rung. Horizontal shunt. | Vertical shunt. Tab Go to the next operand field.
Special Keys
ALT-O Password override. Available only on the Password screen in the configuration software.

E

Logicmaster™ 90-30 Instruction Mnemonics Help (ALT-I)			
Instruction	**Mnemonic**	**Instructions**	**Mnemonic**
Any Contact (search)	&CON	Bit Operation (cont'd)	
-| |-	&NOCON	Shift Left	&SHL
-|/|-	&NCCON	Shift Right	&SHR
<+>---	&CONC	Rotate Left	&ROL
Any Coil (search)	&COI	Rotate Right	&ROR
( )	&NOCOI	Test a Bit	&BT
-(/)-	&NCCOI	Set a bit to 1	&BS
-(↑)-	&PCOI	Set a bit to 0	&BCL
-(↓)-	&NCOI	Locate a bit set to 1	&BP
-(S)-	&SL	Data Move	
(R)	&RL	Move	&MOV
-(SM)-	&SM	Block Move	&BLKM
(RM)	&RM	Block Clear	&BLKC
-(M)-	&NOMC	Shift Register	&SHF
-(/M)-	&NCM	Bit Sequencer	&BI
---<+>	&COILC	Comm Request	&COMMR
Links		Table (Search for)	
----	&HO	Array Move	&AR
|	&VE	Equal	&SRCHE
Timers		Not Equal	&SRCHN
On-Delay Timer	&ON	Greater Than	&SRCHGT
Elapsed Timer Off Delay Timer	&TM &OF	Greater Than/Equal	&SRCHGE
Counters		Less Than	&SRCHLT
Up Counter	&UP	Less Than/Equal	&SRCHLE
Down Counter	&DN	Conversion	
Arithmetic		Convert to Integer	&I_BCD4
Add	&AD	Convert to BCD 4	&BCD4
Subtract	&SUB	Control	
Multiply	&MUL	Call	&CA
Divide	&DIV	Do I/O	&DO
Modulo Divide	&MOD	PID - IND Algorithm	&PIDIS
Square Root	&SQ	Pid - ISA Algorithm	&PIDIN
Relational		End	&END
Equal	&EQ	Comment	&COMME
Not Equal	&NE	Service Request	&SV
Greater Than	&GT	Non-nested MCR	&MCR
Greater Than/Equal	&GE	End non-nested MCR	&ENDMCR
Less Than	&LT	Nested MCR	&MCRN
Less Than/Equal	&LE	End nested MCR	&ENDMCRN
Bit Operation		JUMP	&JUMP
AND	&AN	JUMPN	&JUMPN
OR	&OR	LABEL	&LABEL
Exclusive OR	&XO	LABELN	&LABELN
Invert (NOT)	&NOT		

Instructions with a type modifier, listed in the following table, may have the type modifier appended with and underscore (_) as a separator. For example & ADD_INT

Instruction Type:	Modifier:	Used with:
Signed Integer	_I	Math, Relational, Data Move, & Table
Double Precision Integer	_DI	Math, Relational, & Table
Bit	_BI	Data Move & Table
Byte	_BY	Table
Word	_W	Bit Operation, Data Move, & Table
BCD4	_BCD4	Conversion
Tenths of Seconds	_TEN	Timers
Hundredth of Seconds	_HUN	Timers

Logicmaster™ 90-30 Instruction Mnemonics Help (ALT-I)

Instruction | Mnemonic | Instructions | Mnemonic
Any Contact (search) &CON; -| |- &NOCON; -|/|- &NCCON; <+>--- &CONC; Any Coil (search) &COI; ( ) &NOCOI; -(/)- &NCCOI; -(↑)- &PCOI; -(↓)- &NCOI; -(S)- &SL; (R) &RL; -(SM)- &SM; (RM) &RM; -(M)- &NOMC; -(/M)- &NCM; ---<+> &COILC; Links; ---- &HO; | &VE; Timers; On-Delay Timer &ON; Elapsed Timer &TM; Off Delay Timer &OF; Counters; Up Counter &UP; Down Counter &DN; Arithmetic; Add &AD; Subtract &SUB; Multiply &MUL; Divide &DIV; Modulo Divide &MOD; Square Root &SQ; Relational; Equal &EQ; Not Equal &NE; Greater Than &GT; Greater Than/Equal &GE; Less Than &LT; Less Than/Equal &LE; Bit Operation; AND &AN; OR &OR; Exclusive OR &XO; Invert (NOT) &NOT
Bit Operation (cont'd); Shift Left &SHL; Shift Right &SHR; Rotate Left &ROL; Rotate Right &ROR; Test a Bit &BT; Set a bit to 1 &BS; Set a bit to 0 &BCL; Locate a bit set to 1 &BP; Data Move; Move &MOV; Block Move &BLKM; Block Clear &BLKC; Shift Register &SHF; Bit Sequencer &BI; Comm Request &COMMR; Table (Search for); Array Move &AR; Equal &SRCHE; Not Equal &SRCHN; Greater Than &SRCHGT; Greater Than/Equal &SRCHGE; Less Than &SRCHLT; Less Than/Equal &SRCHLE; Conversion; Convert to Integer &I_BCD4; Convert to BCD 4 &BCD4; Control; Call &CA; Do I/O &DO; PID - IND Algorithm &PIDIS; Pid - ISA Algorithm &PIDIN; End &END; Comment &COMME; Service Request &SV; Non-nested MCR &MCR; End non-nested MCR &ENDMCR; Nested MCR &MCRN; End nested MCR &ENDMCRN; JUMP &JUMP; JUMPN &JUMPN; LABEL &LABEL; LABELN &LABELN

Instructions with a type modifier, listed in the following table, may have the type modifier appended with and underscore (_) as a separator. For example & ADD_INT.

Instruction Type: | Modifier: | Used with:
Signed Integer | _I | Math, Relational, Data Move, & Table
Double Precision Integer | _DI | Math, Relational, & Table
Bit | _BI | Data Move & Table
Byte | _BY | Table
Word | _W | Bit Operation, Data Move, & Table
BCD4 | _BCD4 | Conversion
Tenths of Seconds | _TEN | Timers
Hundredth of Seconds | _HUN | Timers

Logicmaster™ 90-30 Instruction Mnemonics Help (ALT-I)

Instruction | Mnemonic | Instructions | Mnemonic
Any Contact (search) &CON; -| |- &NOCON; -|/|- &NCCON; <+>--- &CONC; Any Coil (search) &COI; ( ) &NOCOI; -(/)- &NCCOI; -(↑)- &PCOI; -(↓)- &NCOI; -(S)- &SL; (R) &RL; -(SM)- &SM; (RM) &RM; -(M)- &NOMC; -(/M)- &NCM; ---<+> &COILC; Links; ---- &HO; | &VE; Timers; On-Delay Timer &ON; Elapsed Timer &TM; Off Delay Timer &OF; Counters; Up Counter &UP; Down Counter &DN; Arithmetic; Add &AD; Subtract &SUB; Multiply &MUL; Divide &DIV; Modulo Divide &MOD; Square Root &SQ; Relational; Equal &EQ; Not Equal &NE; Greater Than &GT; Greater Than/Equal &GE; Less Than &LT; Less Than/Equal &LE; Bit Operation; AND &AN; OR &OR; Exclusive OR &XO; Invert (NOT) &NOT
Bit Operation (cont'd); Shift Left &SHL; Shift Right &SHR; Rotate Left &ROL; Rotate Right &ROR; Test a Bit &BT; Set a bit to 1 &BS; Set a bit to 0 &BCL; Locate a bit set to 1 &BP; Data Move; Move &MOV; Block Move &BLKM; Block Clear &BLKC; Shift Register &SHF; Bit Sequencer &BI; Comm Request &COMMR; Table (Search for); Array Move &AR; Equal &SRCHE; Not Equal &SRCHN; Greater Than &SRCHGT; Greater Than/Equal &SRCHGE; Less Than &SRCHLT; Less Than/Equal &SRCHLE; Conversion; Convert to Integer &I_BCD4; Convert to BCD 4 &BCD4; Control; Call &CA; Do I/O &DO; PID - IND Algorithm &PIDIS; Pid - ISA Algorithm &PIDIN; End &END; Comment &COMME; Service Request &SV; Non-nested MCR &MCR; End non-nested MCR &ENDMCR; Nested MCR &MCRN; End nested MCR &ENDMCRN; JUMP &JUMP; JUMPN &JUMPN; LABEL &LABEL; LABELN &LABELN

Instructions with a type modifier, listed in the following table, may have the type modifier appended with and underscore (_) as a separator. For example & ADD_INT

Instruction Type: | Modifier: | Used with:
Signed Integer | _I | Math, Relational, Data Move, & Table
Double Precision Integer | _DI | Math, Relational, & Table
Bit | _BI | Data Move & Table
Byte | _BY | Table
Word | _W | Bit Operation, Data Move, & Table
BCD4 | _BCD4 | Conversion
Tenths of Seconds | _TEN | Timers
Hundredth of Seconds | _HUN | Timers

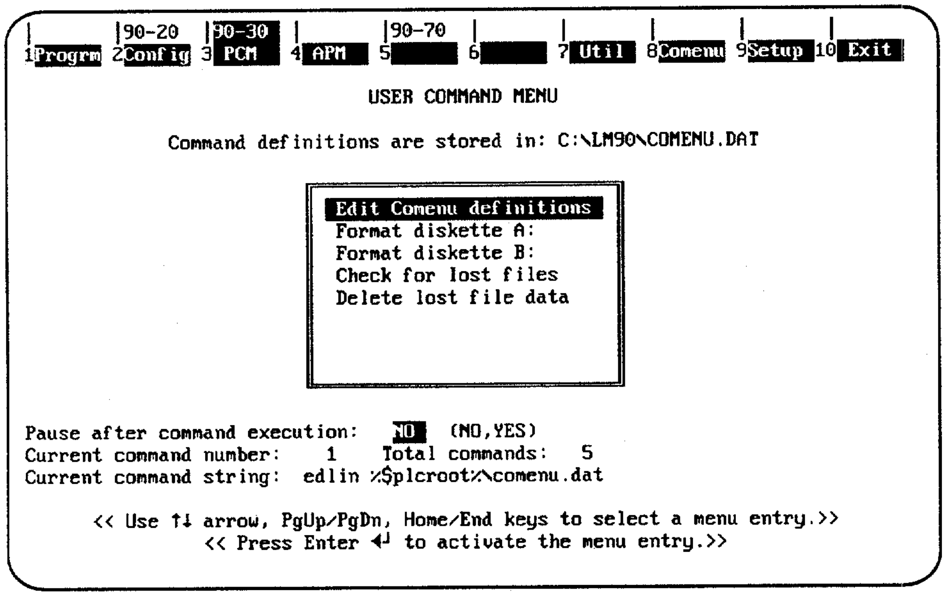
Appendix F User Command Menu

You can maintain a file of MS-DOS executable commands outside of the Logicmaster software packages. This command definition file named COMENU.DAT contains a list of task entries, each with a unique display label and associated MS-DOS command. The file is stored in the Logicmaster home directory (normally **\LM90**) and can be edited using an MS-DOS compatible text editor such as EDLIN. The MS-DOS commands in this file can be used to run other software packages and perform routine MS-DOS functions, such as disk maintenance.

Accessing the User Command Menu

The MS-DOS executable commands maintained in the COMENU.DAT file are accessed from the User Command menu called *Comenu.*

1. To display this menu, press **Comenu (F8)** from the menu of Series 90 PLCs and functions. A sample Comenu is shown below.



|90-20 |90-30 | |90-70
1Progrm 2Config 3 PCM 4 APM 5 6 7 Util 8Comenu 9Setup 10 Exit

```
                         USER COMMAND MENU

          Command definitions are stored in: C:\LM90\COMENU.DAT

                     Edit Comenu definitions
                     Format diskette A:
                     Format diskette B:
                     Check for lost files
                     Delete lost file data

Pause after command execution:   NO   (NO,YES)
Current command number:    1    Total commands:   5
Current command string:  edlin %$plcroot%\comenu.dat

     << Use ↑↓ arrow, PgUp/PgDn, Home/End keys to select a menu entry.>>
              << Press Enter ↵ to activate the menu entry.>>
```

2. When **F8** is pressed, the Logicmaster startup program reads the Comenu definition file and creates a display similar to the one shown above.
3. Use the cursor keys, the **Page Up** and **Down** keys, or the **Home** and **End** keys to select a command. The label for the currently selected entry is shown in reverse video. If more command definitions are available than can be displayed on the screen, use the cursor keys to scroll through the entries. The number of the currently selected command and the total number of commands are displayed beneath the labels window. The MS-DOS command string for the current entry is shown beneath the labels window for verification of the command text associated with the entry.
4. Move the cursor to the desired command label and press the **Enter** key. The menu program will clear the screen and submit the selected command to MS-DOS for execution. Subsequent screens are controlled by MS-DOS and the selected command.
5. When the command is completed, control will normally return immediately to the Comenu display. However, you may wish to pause after the execution of the command in order to study the display generated by the command. If a pause is desired, enter **Y** (Yes) in the *Pause after command execution* field, or use the **Tab, Shift-Tab**, **Space**, or **Backspace** key to toggle the value of the field from **No** to **Yes**.
6. After the pause, MS-DOS will prompt you to press any key in order to continue. When a key is pressed, the User Command menu is displayed again.
7. Press **Exit (F10)** or the **Escape** key to return to the menu of Series 90 PLCs and functions.

Creating a COMENU.DAT file

A default command definition file is provided with the Logicmaster 90-30/20/Micro software. The default file provides example definitions and includes entries for performing disk formatting.

Definitions may be entered in the COMENU.DAT file in either of two forms:

```
<display label> = <DOS command string>
```

or

```
<display label> =
<DOS command string>
```

according to these guidelines:

1. Blank lines and lines beginning with a semicolon are ignored.
2. Lead and trailing space and tab characters around the label and command are ignored.
3. The label may be any text up to 50 characters long. Labels which exceed this length are truncated.
4. The equal sign (=) must appear on the same line as the label. A carriage return and line feed may follow the equal sign if you want the command text to start on a separate line.

5. Each command string is terminated by a carriage return and line feed, or by the end-of-file. The command string itself should not contain any carriage returns, even if the string is longer than can be displayed on a single line.
6. The command string is limited by MS-DOS to a maximum length of 127 characters. Commands which exceed this length are ignored, and the associated label is not displayed.
7. A maximum of 100 definitions can be processed; any additional definitions beyond the limit are ignored.
8. You may include in the definition file a command for editing the definition file itself. This will allow you to modify the file even while remaining within the Logicmaster environment.

Example Comenu Definition File

The following is an example Comenu definition file used to create the Comenu shown above.

```
; This is the Comenu definition file.
;   Blank lines and lines starting with a semicolon are ignored.

; The following entry may be used to start the editor on this file,
; so that additional revisions may be made without leaving the LM90 shell.

Edit Comenu definitions =
    edlin  \LM90\COMENU.DAT

; Disk utilities:

Format diskette A:     = format a: /v
Format diskette B:     = format b: /v
Check for lost files   = chkdsk /f
Delete lost file data  = del \file0*.chk
Duplicate floppy A:    = diskcopy a: a:

; Third-party packages:

Cadepa grafcet package = \SWN\CADEPA

; User applications:

Application 1 = \appl\appl.bat
Application 2 = \appl\app2.bat
```

Appendix G Files Created with Logicmaster 90-30/20/Micro Software

This appendix lists the files created with Logicmaster 90-30/20/Micro software. The files are identified here by their extensions. The content of each file is briefly discussed, along with an explanation of when the file is created. None of these files is created during the INSTALL process.

Caution

Do not use MS-DOS to copy individual files from one folder to another or to delete files. Doing so may produce unexpected results. MS-DOS may only be safely used to copy an entire program folder to another program folder of the same name.

Files in the Program Folder

The following files are associated with program logic. They are created as a result of editing the program.

Note

.PDT and .STE are new file extensions available with Release 3.50 of Logicmaster 90-30/20 software. .PDT combines .PRG and .DAT files from earlier releases into one file. .STE combines .SYM and .NXP files from earlier releases into one file.

Extension	File Description	When Logicmaster 90-30/20/Micro Software Creates the File
.PDT	This file contains the program logic.	The _MAIN.PRG file is created during the first edit session in the program. When a subroutine is declared, its .PDT file is created.
.DEC	This file contains program block declarations, as well as other data including information about the coil usage and retentive sense of the %Q and %M references.	The _MAIN.DEC file is created during the first edit session in the program. There is one .DEC file for the entire program.
.STE	This file contains the nicknames, reference descriptions, and identifiers associated with a particular subroutine.	The _MAIN.STE file is created during the first edit session. The .STE file for a subroutine is created when a nickname, reference description, or identifier is first defined within that subroutine.
.EXP	This file contains the text for comment rungs.	The .EXP file is created when the first comment rung for the program is programmed.
.LH1	This file contains the header from each .PDT file.	The .LH1 file is created during the first edit session. There is one .LH1 file per folder.
.SDE	This file contains material that was selected and written to a program segment.	The .SDE file is created when you press **Write (F5)**. Unlike the other files listed above, .SDE files may be written to any MS-DOS directory. The default destination is the current folder.

When specifying values using the Reference table function, Logicmaster 90-30/20/Micro software creates files to contain this information. The folder may contain these two files for each reference type. Only reference types which permit overrides may have override files. (The %I reference is used here only as an example.)

Extension	File Description	When Logicmaster 90-30/20/Micro Software Creates the File
.I	The _MAIN.I file contains initial values for %I references.	The _MAIN.I file is created when data is first entered for any %I reference. There is only one .I file per folder.
.IO	The _MAIN.IO file contains initial override data for %I references.	The _MAIN.IO file is created when data is first entered for any %I reference. There is only one .IO file per folder.

When specifying formats using the reference table function, Logicmaster 90-30/20/Micro software creates files to contain this information.

Extension	File Description	When Logicmaster 90-30/20/Micro Software Creates the File
.RDF	The .RDF files contain information specifying the formats for reference data.	The FIXED.RDF and MIXED.RDF files are created when a reference format is specified for any of the fixed or mixed tables, respectively. A folder may contain up to two of these files.

The configurator packages creates files to contain the configuration data.

Extension	File Description	When Logicmaster 90-30/20/Micro Software Creates the File
.CFG	The IOCFG.CFG file contains information about the I/O configuration for the program. The CPUCFG.CFG file contains information about the CPU configuration for the program.	The IOCFG.CFG and CPUCFG.CFG files are created when any configuration data is first edited.

Teach files are used to store a sequence of keystrokes that may be repeated.

Extension	File Description	When Logicmaster 90-30/20/Micro Software Creates the File
.DEF	This file contains keys that were pressed while in Teach mode.	The .DEF file is created when Teach mode is entered and a file name is specified.

Files in the Logicmaster 90-30/20/Micro Home Directory

For your convenience, Logicmaster 90-30/20/Micro software creates files that store information to be used as defaults the next time the software is booted.

Extension	File Description	When Logicmaster 90-30/20/Micro Software Creates the File
.DAT	The modem auto dial feature writes information in the MODEM.DAT file. This file is only used with Logicmaster 90-30 software.	The .DAT file is created using Logicmaster 90-30 software when modem information is saved.
.PSU	These files contain setup information that is specified using the Programmer Setup functions. A file name may be specified for these setup files. Default file names are %COM1.PSU or %COM2.PSU, %PLC030.PSU, and %WSI030.PSU. These files contain information that can be used as defaults on the Printer Serial Port Setup, Select SNP Connections, and Programmer WSI Serial Port Setup screens, respectively.	The .PSU files are created when Save (F7) is pressed.
.SET	This file contains information that is specified using the Print functions.	A PRINT.SET file is created when information on the Setup Printer Parameters screen is saved. The SCRPRINT.SET file is associated with information on the Select Screen Print Destination screen.
.FLD	The LAST30.FLD file contains the name of the selected folder.	The .FLD file is created when the first folder is created when booting the software. It is used as the default the next time the software executes.

Cross Reference Data Files

Print function cross reference data files are not automatically deleted when you exit the print function. You can, however, specify that the files be deleted at the end of a cross reference listing by setting the *Delete Files After Use* field on the Cross Reference screen to **Y** (Yes). (Refer to chapter 9, section 3, "Print Program," for more information.)

Extension	File Description	When Logicmaster 90-30/20/Micro Software Creates the File
.XRF	The .XRF file contains cross reference data for a single logic block.	A .XRF file is created for each logic block when printing a listing containing cross reference tables, reference use tables, or in-ladder cross references.
.XOV	The PRINT.XOV file contains data for managing the .XRF files.	A PRINT.XOV file is created when the all blocks option is selected while printing a listing containing cross reference tables, reference use tables, or in-ladder cross references.

Appendix H Common User Errors

Error Message: "Error Detected in WSI Board"

Some users of IBM-clone computers have experienced difficulty using Logicmaster 90-30/20/Micro software. The computer backplane clock rate for the WSI Board slot should not be greater than 8 mHz.

In addition, certain device drivers installed by CONFIG.SYS when DOS is booted can prevent the expansion bus from performing 8-bit data transfers with the Work Station Interface (WSI) Board. When this situation occurs, the Logicmaster 90-30/20/Micro software displays the error message, "Error detected in WSI Board," during initialization.

To correct the problem for a COMPAQ computer:

1. Copy CONFIG.SYS to CONFIG.SAV to preserve the original version of the CONFIG.SYS file.
2. Examine the CONFIG.SYS file (type CONFIG.SYS) for a DEVICE statement similar to one of these statements:

```
device=xxxxxxx.sys bus 16
device=c:xxx/sys bus 16
device=c:xxx.sys <other parameters> bus 16
```

   where *<other parameters>* represents any number of words or numbers separated by spaces.

3. Using a text editor such as EDLIN, remove "bus 16" from the statement.
4. Save the modified version.
5. Power the computer off and on. A warm boot (i.e., pressing CTRL-ALT-Delete) will not reset the hardware for correct 8-bit operation.

If some other application program requires the unmodified form of the device driver, different versions of the CONFIG.SYS file can be used for it and for Logicmaster 90-30/20/Micro software. A good method for managing this technique is to use a separate batch file to copy each version when it is needed from its permanent, distinctive file name (e.g., CONFIG.LM) to CONFIG.SYS.

Trouble Communicating with the Series 90-30 or Series 90-20 PLC
Error Message: "Error Detected in WSI Board Port"
Error Message: "Error Loading Code into WSI Board"

Certain boards installed in the computer can conflict with the interrupt used by the Work Station Interface Board. The WSI Board uses the expansion bus IRQ3 interrupt request line. IRQ3 is assigned to the COM2 serial port and is also used by other communication adapters. If the WSI Board is installed in a computer where COM2 is used by an application which requires the use of interrupts (e.g., serial mouse), then neither the WSI Board nor the COM2 application will operate properly. The WSIB2 Board has a software-configurable interrupt request line. Refer to the information on PLC communications options in appendix C, *Programmer Environment Setup,* for information on moving the interrupt request line to one that does not conflict.

The WSI Board uses the MS-DOS memory addresses ranging from CE00:0 to CE00:1FFF. Other applications must avoid this address range. It may be possible to configure the application to use a different address range than that of the WSI Board. Consult the documentation for the conflicting application for more information.

The WSI Board uses the I/O port addresses 310 hex to 313 hex. Other applications must avoid these port addresses in order to prevent conflicts.

Error Message: "File System Error"

If the message, "File System Error", occurs, the FILES allocated in the CONFIG.SYS file is probably too low. Change or add this line to the CONFIG.SYS file:

```
FILES=20
```

You must then reboot your computer in order to activate the new value for FILES in the modified CONFIG.SYS file. To reboot your computer, press CTRL-ALT-Delete.

If this line already exists, try changing the number to 30 (i.e., FILES=30) and rebooting your computer.

Error Message: "Read/Write PLC Initialization Aborted"

If the message, "Read/Write PLC Initialization Failure", occurs while attempting to run Logicmaster 90-30/20/Micro software, the WSI Board may not be seated properly. Re-install the WSI Board in the computer slot or try another slot. If this does not correct the problem, the board may be bad.

Error Message: "Constant Out of Range"

If the message, "Constant Out of Range" occurs while trying to enter hexadecimal constants, you may be trying to enter hex constants into integer type functions. Use the Types (F10) function softkey to change the function type.

Cannot Install the Software

You must have at least 1.4 Megabytes of hard disk space and an additional 600K bytes of temporary hard disk space if Lotus/Intel/Microsoft expanded memory (LIM 3.0 or higher) is not available for Logicmaster 90-30/20 software. Also, be sure the CONFIG.SYS file has files set to at least 20, i.e., FILES=20.

If the message, "The unarchive program failed (PK11)", occurs, check the CONFIG.SYS file. It should have the line FILES=20. If it does, try changing that to FILES=30, reboot the computer, and try the INSTALL procedure again.

If the message, "The unarchive program failed (PK1)", occurs, try installing the software again. If the INSTALL procedure still fails, the INSTALL diskette is probably bad.

Error Message: "Comm Driver Not Loaded"

If the message, "Comm Driver Not Loaded" occurs, you may be trying to run the standard serial communications version of Logicmaster 90-30/20/Micro software without the correct version of MS-DOS and/or without a memory manager. Refer to chapter 1, "Introduction," for a list of what you will need in order to run the Logicmaster 90-30/20/Micro software.

Printer Output is Garbled

If the *Specify Graphics Printer* option was enabled in the Logicmaster 90 setup package, make sure that the attached printer supports the IBM graphics character set.

If you have a serial printer, make sure the appropriate serial port has been configured with the MS-DOS mode command to match the printer settings. The MS-DOS mode command should be used to configure the serial printer port before starting up the Logicmaster 90 software package. For more information, see chapter 6, "Programmer Setup."

Error Message: "Port/File Access Denied"

Attempting to print logic to a virtual drive which is full results in the error "Port/File access denied." To correct the problem, delete some files from your virtual disk.

System Software Error ID: 0000 EX: 0000

Logicmaster 90-30/20/Micro software does not have enough available RAM to perform the operation. Check the AUTOEXEC.BAT and CONFIG.SYS files to remove any device drivers or Terminate and Stay Resident (TSR) programs in order to free more RAM.

Busy Message Displayed After Load

The message, "Loading . . ." will continue to be displayed after a load operation fails because of incompatible files. To clear the busy message, press any key.

Appendix I Variable Declaration Table Import/Export Using Comma Separated Variable (CSV) Format

Comma Separated Variable (CSV) Format and SNF Format

The Shared Name File (or SNF) format is an extension of the industry-standard Comma Separated Variable (CSV) format. If you are planning on importing a file into the Variable Declarations Table as discussed on page 3-41, you must ensure that your file is in SNF format. When exporting Variable Declarations as discussed on page 3-43, Logicmaster places the files into SNF format (see page 3-41 for more information on SNF format).

SNF files contain two types of information, Comments and Records. Comment lines begin with the Comment Line Sequence ## (i.e., two consecutive pound characters as the first non-blank characters in the line). Records, except for Header Section records (see below), do not begin with the Comment Line sequence.

General SNF Format Rules

- Shared Name Files shall have a default extension of ".SNF". The recommended naming convention is that the filename be the same as the source file (folder) from which the SNF is generated.
- Comment lines begin with the double pound sign ("##") character sequence as the first non-blank character.
- Comment lines may appear anywhere within the SNF, but may not occur inside a record.
- Records continued over several lines may not have intervening comment lines.
- The SNF will consist of three sections which must occur in order:

  (1) an optional Header section

  (2) a required Field Names section

  (3) a required Data section.
- All sections are made up of comma separated variable (CSV) records. Each CSV record is composed of a list of fields separated by commas. Individual fields may be enclosed within double quotation marks, and fields that contain commas must be enclosed within quotes.
- All records in a section must include the same fields, and the fields must be in the same order in each record. Fields in records may be left empty, but empty fields must be delimited with commas like all other fields.

- Spaces surrounding the comma delimiters in a record are ignored. To include leading or trailing spaces in a field, surround the field with quotation marks.
- Records may span one or more lines of text in an SNF file. A line of text in an SNF file ends with the new line character (or character sequence) appropriate for the operating system. The last line of text in the file must include a new line character.
- Any line beginning with a double colon sequence (::) is considered a continuation of the record started on the previous line. Fields must be contained within a single line, and their trailing comma delimiter must follow the field on the same line as the field (i.e., lines of text followed by continuation lines must end in a non-quoted comma).
- Blank lines are not allowed.
- The maximum line size is 2000 characters.
- The maximum record size is 2000 characters not counting continuation characters and new line characters.
- The maximum number of fields in a record is 100.

Header Section Rules

- Header section records may not span multiple lines.
- The Header section contains structured comments which provide some context for the file.
- All header section entries must be accepted by any package using the SNF format, but they need only interpret and use the ones they recognize.

Field Names Section Rules

- The Field Names Section contains one and only one record.
- The Field Names Section begins with the first non-comment line in the file.
- Field Names are case insensitive.
- PT_ID must be the first field in the Field Names section.
- No empty field are permitted in the Field Names section.

Data Section Rules

- The Data Section follows the Field Names Section.
- The Data Section begins on the first non-comment line after the Field Name section and continues to the end of the file.
- Predefined Keywords in the Data section are case insensitive.
- PT_IDs in the Data Section are case insensitive.
- Empty fields are permitted in the Data section, but they must be delimited by commas.
- If the PT_ID field is empty, the ADDR field must be present and not empty.
- The format of numbers shall be whatever Excel 5.0 uses as a default when creating a CSV files.

The Header Section

This section of the SNF contains fields which define the context of the SNF. The Header section has four keywords currently defined. It is expected that additional keywords will be defined by GE Fanuc and in addition customers will wish to add their own keywords.

##&&Creator	This identifies the program which specified the content of this SNF file. Three possible values generated by GE Fanuc programs are: "CIMPLIC-ITY", "LOGICMASTER", or "PROCESS_90". Other values for this field will not cause an error.
##&&CreateDate	This specifies the date when this SNF file was created. The date format shall be in the format "(d)d-mmm-yyyy", for example, 31-MAY-1994, 4-NOV-1986, or 11-Sep-1994
##&&ProgDate	The date of the creation of the program with which this SNF file is associated. If Logicmaster is creating the SNF file, this field contains the date the program was last updated. If the PROCESS 90 Editor is creat-ing the SNF file, this field contains the date the associated Strategy was last updated.
##&&FileType	This identifies what type of data is contained within this SNF. The pos-sible values for this field are: "STRATEGY", "I/O", "I/O_WITH_FAULTS", "ALGORITHM_DATA".

The Field Names Section

This section of the SNF is used to specify the names of the fields in the Data section, and the order of these named fields. The order of the fields in the field names section is the order to be used in all records of the Data section. The field names recognized by Logicmaster are shown below.

Note: Cimplicity® recognizes these field names plus field names specified in the *Cimplicity® Systems Import/Export Utility Operation Manual* (GFK-0923).

Field Name	Meaning of This Field
PT_ID	Tag Name of this point *
ADDR	A mnemonic reference to a PLC memory location.
DESC	An ASCII string describing what data this point represents.
PT_TYPE	What type of data is being described by this point

There are additional field names used by Cimplicity and by PROCESS90. For those field names refer to the documentation for each product. Additional fields which are not supported by the importing program will be ignored without error.

The field name order as shown in the above table is the preferred order.

Example CSV (SNF) File

The following is an example of the CSV format (i.e., SNF format) used by Logicmaster. This example was exported from Logicmaster as noted in the first four lines. The double pound sign and double ampersand denotes commented lines. If you had created this in a spreadsheet program, it would look the same without the commented lines.

```
##&&Creator,LOGICMASTER
##&&CreateDate,17-MAR-1995
##&&ProgDate,21-FEB-1995
##&&FileType,I/O
PT_ID,ADDR,DESC,PT_TYPE
D_HATCH,%I00001,Dry Hatch Status,BOOL
W_HATCH,%I00002,Wet Hatch Status,BOOL
A_RESET,%I00008,Alarm  Reset,BOOL
ALM_ACK,%Q01000,Alarm Acknowledge from BCS,BOOL
BCS_INT,%Q01001,BCS initialization,BOOL
NEXT_BT,%Q01002,Next Batch,BOOL
AGIT_ON,%M01001,Agitator Status,
BULK_ON,%M01002,Bulk Ingred Valve Status,
STEAM_O,%M01003,Steam Valve Status,
AGIT_SP,%R00011,Agitate Speed Setpnt,INT
TEMP_SP,%R00012,Reactor Temp   Setpnt,INT
AGIT_T,%R00013,Agitate Time  (seconds),INT
BULK_AD,%R00014,Bulk   Add    Measure (lbs),INT
PHCTL1A,%R00021,BulkAddPhase  Control Reg #1,INT
PHCTL1B,%R00022,BulkAddPhase  Control Reg #2,INT
PH_CB1A,%R00023,BulkAddPhase  Ctl But Reg #1,INT
PH_CB1B,%R00024,BulkAddPhase  Ctl But Reg #2,INT
INTL_R1,%R00025,BulkAddPhase  Intlock Reg,INT
PHSTS_1,%R00026,BulkAddPhase  Status  Reg,INT
BK_WGT,%R00027,Bulk   Added  Weight,INT
R_TEMP,%AI0001,Reactor Temp,INT
LIQ_LEV,%AI0002,ReactorLiquid Level,INT
AGIT_S,%AI0003,Agitate Speed,INT
ST_F_RT,%AI0004,Steam  Flow   Rate,INT
AGIT_SC,%AQ0001,Agitate Speed Command,INT
ST_V_PO,%AQ0002,Steam  Valve  Position,INT
```

GE Fanuc Automation North America, Inc., Charlottesville, Virginia