

UNION SWITCH & SIGNAL 

A member of the ANSALDO Group
5800 Corporate Drive Pittsburgh, PA 15237

SERVICE MANUAL 6300A

Non-Vital Application Logic Programming

GENISYS[®]

NON-VITAL LOGIC EMULATOR

MICROLOK PLUS[™]

VITAL + NON-VITAL CONTROL PACKAGE

(NON-VITAL SECTION)

Up to and including:

Executive Software Revision 11

Application Logic Software Version 3.0

October, 1991
A-10/91-2645-1
ID0312F, 0313F

COPYRIGHT 1991, UNION SWITCH & SIGNAL INC
PRINTED IN USA

ANSALDO
Trasporti

Revised and new pages of this manual are listed by page number and date:

Page No.

Date

Page No.

Date

CONTENTS

<u>Section</u>	<u>Page</u>
I INTRODUCTION TO MANUAL	1-1
1.1 PURPOSE AND ARRANGEMENT	1-1
1.2 FAMILY OF MANUALS	1-1
II GENERAL INFORMATION - GENISYS	2-1
2.1 INTRODUCTION	2-1
2.1.1 Overall System	2-1
2.1.2 Application and Executive Software	2-1
2.2 COMPONENTS	2-3
2.2.1 Cardfile	2-3
2.2.2 Printed Circuit Boards	2-5
2.2.2.1 Controller	2-5
2.2.2.2 Relay-Output PCBs	2-5
2.2.2.3 Optical-Input PCBs	2-5
2.2.2.4 Power Supply Converter PCBs	2-5
2.3 NON-VITAL SECTION SPECIFICATIONS (Programming Related)	2-6
III GENERAL INFORMATION - MICROLOK PLUS	3-1
3.1 INTRODUCTION	3-1
3.1.1 Overall System	3-1
3.1.2 Non-Vital Section	3-1
3.1.3 Application and Executive Software	3-3
3.2 COMPONENTS	3-4
3.2.1 Cardfile	3-4
3.2.2 PCBs	3-4
3.2.2.1 Controller N451441-5602	3-4
3.2.2.2 Relay-Output PCBs	3-4
3.2.2.3 Optical-Input PCBs	3-4
3.3 SPECIFICATIONS (Programming Related)	3-7/8
IV PROGRAMMING PROCEDURES - GENISYS AND MICROLOK PLUS	4-1
4.1 GENERAL	4-1
4.2 PROGRAMMING LANGUAGE	4-1
4.2.1 Terms	4-1
4.2.1.1 Character Set	4-1
4.2.1.2 Reserved Words	4-2
4.2.1.3 User-Defined Symbols	4-2
4.2.1.4 Delimiters	4-3
4.2.2 Formats	4-3
4.2.2.1 General Arrangement of Statements	4-3
4.2.2.2 Non-Program Comments	4-3
4.2.2.3 Compiler Switches	4-4
4.2.3 Program Examples	4-8
4.2.3.1 Local Input/Output	4-8
4.2.3.2 Internal Relays and Stick Logic	4-9
4.2.3.3 Timing Relays	4-10
4.2.3.4 Master/Slave Communications	4-11
4.2.4 Detailed Statement Descriptions	4-12
4.2.4.1 "PROGRAM" Statement	4-12
4.2.4.2 "INTERFACE" Section	4-13

CONTENTS (Cont'd)

<u>Section</u>		<u>Page</u>
4.2.4.3	"VAR" Section	4-16
4.2.4.4	"TIMER" Section	4-16
4.2.4.5	Main Program Body	4-18
4.2.4.6	ASSIGN Statement	4-18
4.2.5	Run Time System Description	4-20
4.2.5.1	Input/Output Description	4-20
4.2.5.2	Logic Processing	4-20
4.2.5.3	Serial Communications - Pre-Defined Relays	4-22
4.2.5.4	Valid Bit Option - Introduction	4-23
4.2.6	Relay Models and Programming Techniques	4-24
4.3	GENISYS DEVELOPMENT SYSTEM (G.D.S.) - GENERAL	4-26
4.4	G.D.S. - AVAILABLE FILES	4-26
4.5	G.D.S. - COMPILER	4-27
4.6	G.D.S. - SIMULATOR	4-28
4.6.1	General	4-28
4.6.2	Access to Simulator	4-29
4.6.2.1	General	4-29
4.6.2.2	Procedure	4-29
4.6.3	Standard Formats	4-30
4.6.4	Simulator Operation	4-31
4.6.4.1	General	4-31
4.6.4.2	Sample Program	4-31
4.6.4.3	Help Screen	4-35
4.6.4.4	Display IO Command	4-36
4.6.4.5	Display Triggers Command	4-36
4.6.4.6	Display Relays Command	4-38
4.6.4.7	Remove Command	4-39
4.6.4.8	Input Command	4-39
4.6.4.9	Relay Set and Clear Commands	4-40
4.6.4.10	Increment Command	4-45
4.6.4.11	Display Timers Command	4-46
4.6.4.12	Execute Command	4-47
4.6.4.13	Trace Command	4-47
4.6.4.14	Run Command	4-48
4.6.4.15	Value Command	4-50
4.6.4.16	Read Command	4-50
4.6.4.17	Print Command	4-51
4.6.4.18	Reset and Quit Commands	4-53
4.6.4.19	Color CRT Commands	4-53
4.7	G.D.S. - EPROM SIZE ESTIMATES PROGRAM	4-54
4.7.1	General	4-54
4.7.2	Sample Execution	4-55
4.8	G.D.S. - EPROM PROGRAMMER	4-55
4.8.1	General	4-55
4.8.2	Initial Configuration File - G.D.S. Versions 1.04 and Higher	4-56
4.8.3	Programmer Operation - G.D.S. Versions 1.01 and Higher	4-57
4.8.4	Error Messages - G.D.S. Versions 1.01 and Higher	4-59
4.8.5	Communications Interrupt	4-62

CONTENTS (Cont'd)

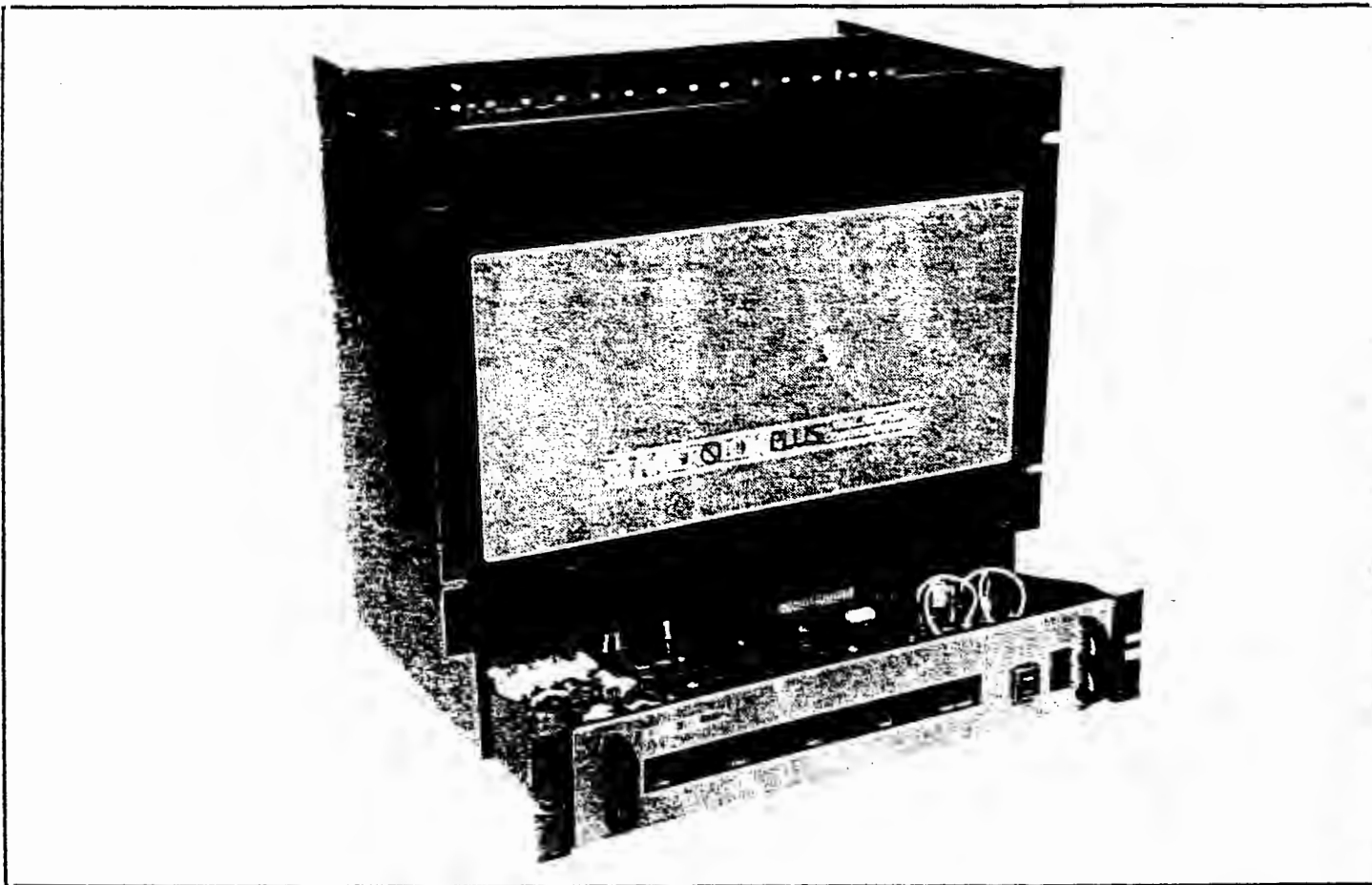
<u>Section</u>		<u>Page</u>
4.8.6	Initial Configuration File - G.D.S. Versions 1.00 through 1.03	4-62
4.8.7	Programmer Operation - G.D.S. Version 1.00	4-62
4.8.8	Error Messages - G.D.S. Version 1.00	4-64
4.8.9	EPROM Programmer Driver - Color Display	4-64
V	MISCELLANEOUS PROGRAM DESIGN NOTES - GENISYS AND MICROLOK PLUS	5-1
5.1	LOGIC AND TIMING OVERFLOWS	5-1
5.2	TIMING ELEMENTS	5-1
5.2.1	Introduction	5-1
5.2.2	General Processing	5-1
5.2.3	Parameters	5-1
5.2.4	Skew Time	5-2
5.3	VALIDATION OPTION	5-3
5.3.1	Introduction	5-3
5.3.2	Parameters	5-3
5.3.3	Recommendations	5-3
5.4	LOGIC QUEUING AND EXECUTION	5-5
5.4.1	Comparison of Hardware and Software Relay Logic	5-5
5.4.2	Breaks Before Makes Rule	5-5
5.4.3	Queuing Options	5-6
VI	MISCELLANEOUS APPLICATION INFORMATION - GENISYS AND MICROLOK PLUS	6-1
6.1	LOCAL I/O	6-1
6.1.1	Using Slave Units as I/O Processors	6-1
6.1.2	Determining the Control Delivery Time	6-1
6.1.2.1	Introduction	6-1
6.1.2.2	Selection Considerations	6-1
6.2	SERIAL COMMUNICATIONS TIMING	6-2
VII	SERIAL COMMUNICATIONS PROTOCOL - GENISYS AND MICROLOK PLUS	7-1
7.1	INTRODUCTION	7-1
7.1.1	Message Format	7-1
7.1.2	Message Sequence	7-2
7.1.3	Good and Bad Messages	7-4
7.2	DETAILED DESCRIPTION	7-6
7.2.1	General Specifications	7-6
7.2.2	Master to Slave Data Transmission	7-7
7.2.2.1	Poll Command	7-7
7.2.2.2	Acknowledge Data Command	7-7
7.2.2.3	Control Command	7-8
7.2.2.4	Recall Indications Command	7-8
7.2.2.5	Execute Controls Command	7-8
7.2.2.6	Common Control Mode	7-8
7.2.3	Slave to Master Data Transmission	7-9
7.2.3.1	Acknowledge Master Response	7-9
7.2.3.2	Indication Data Response	7-9
7.2.3.3	Control Checkback Command	7-10
7.2.4	Control Code Summary	7-10

CONTENTS (Cont'd)

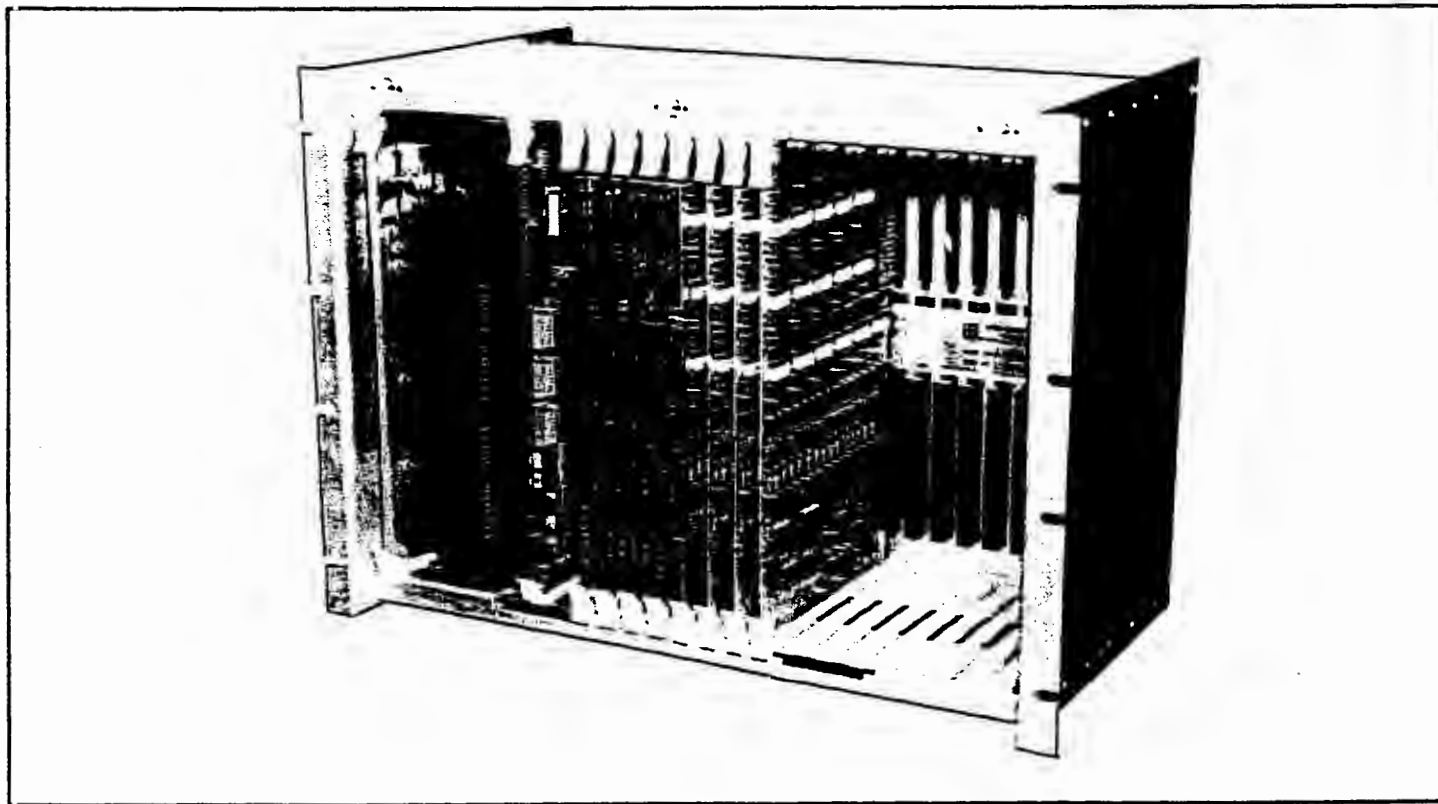
<u>Section</u>	<u>Page</u>
VII SUPPLEMENTAL DATA - GENISYS AND MICROLOK PLUS	8-1
8.1 TOKEN AND PARSING ERROR/WARNING MESSAGES	8-1
8.2 SEMANTIC ERROR MESSAGES	8-2
8.3 CODE SYSTEM PRE-PROGRAMMED EPROMS	8-4
8.4 CONTROLLER PCB HARDWARE PROGRAMMING	9-5
8.4.1 Slave Port Baud Rate (SW1)	8-5
8.4.2 Control Delivery Time (SW2)	8-7
8.4.3 Carrier Mode (SW3)	8-7
8.4.4 Slave Station Address (SW5)	8-8
8.4.5 Key-On and Key-Off Delays (SW6)	8-8
8.4.6 Communications Mode Select (Jumpers)	8-9
8.4.7 Serial Port Test (SW7)	8-8
8.4.8 Serial Port Data Byte Format (SW7)	8-10
APPENDIX A PARTS LIST (DEVELOPMENT SYSTEM EQUIPMENT)	

ILLUSTRATIONS

<u>Figure</u>	<u>Page</u>
2-1 Basic GENISYS System	2-2
2-2 GENISYS Cardfile PCB Arrangement	2-4
3-1 Basic MICROLOK PLUS System	3-2
3-2 MICROLOK PLUS Application and Executive Software	3-4
3-3 MICROLOK PLUS Cardfile PCB Arrangement	3-6
4-1 Master/Slave Communications Programming Reference Diagram	4-15
4-2 ASSIGN Operators, Order of Precedence Samples	4-19
4-3 Queuing Option Reference Diagram	4-21
4-4 Conceptual Relay Models for GENISYS and MICROLOK PLUS Programming	4-24
4-5 Development System Block Diagram	4-26
5-1 Example of Front and Back Contact Assignments	5-5
5-2 Queuing Option Example	5-6
8-1 Controller PCB Manually Selected Options	8-6



MICROLOK® PLUS Cardfile (Cover closed, power drawer open)



GENISYS® Cardfile (Cover removed)

SECTION I
INTRODUCTION TO MANUAL

1.1 PURPOSE

This manual provides instructions for programming the non-vital application software of both the GENISYS Non-Vital Logic Emulator (NVLE) and the non-vital section of the MICROLOK-PLUS Vital + Non-Vital Control Package. These systems share identical non-vital logic and interface circuit boards, as well as executive and application software. PCB hardware and software design revisions affect both systems.

1.2 FAMILY OF MANUALS

This manual is one of eight manuals that cover the GENISYS Non-Vital Logic Emulator, the MICROLOK Vital Interlocking Control System and/or the MICROLOK PLUS Vital + Non-Vital Control Package. The following table summarizes these manuals:

<u>SM No.</u>	<u>System(s) Covered</u>	<u>Purpose</u>
6300A	GENISYS, MICROLOK PLUS	<u>Both Systems:</u> Programming of Non-Vital Application Logic
6300B	GENISYS, MICROLOK PLUS	<u>GENISYS:</u> Hardware Installation, Local and Serial Data Interfacing, Field Troubleshooting <u>MICROLOK PLUS:</u> Local/Serial Data Interfacing and Field Troubleshooting of Non-Vital Section (refer to SM-6400B for MICROLOK PLUS hardware installation)
6300C	GENISYS, MICROLOK PLUS	<u>Both Systems:</u> Shop Troubleshooting of Non-Vital Printed Circuit Boards
6301	GENISYS, MICROLOK PLUS	<u>Both Systems:</u> Installation of GENISYS Development System (G.D.S.) - Hard and Dual-Floppy Disks
6400A	MICROLOK, MICROLOK PLUS	<u>Both Systems:</u> Programming of Vital Application Logic
6400B	MICROLOK, MICROLOK PLUS	<u>MICROLOK:</u> Hardware Installation, Power and Data Interfacing <u>MICROLOK PLUS:</u> Hardware Installation, Power Interfacing, Data Interfacing of Vital Section (refer to SM-6300B for non-vital data interfacing)

<u>SM No.</u>	<u>System(s) Covered</u>	<u>Purpose</u>
6400C	MICROLOK, MICROLOK PLUS	<u>MICROLOK:</u> Field Troubleshooting <u>MICROLOK PLUS:</u> Field Troubleshooting of Vital Section
6401	MICROLOK, MICROLOK PLUS	<u>Both Systems:</u> Installation of MICROLOK Development System (M.D.S.) - Hard and Dual-Floppy Disks

SECTION II GENERAL INFORMATION - GENISYS

2.1 INTRODUCTION

2.1.1 Overall System (See Figure 2-1)

The GENISYS Non-Vital Logic Emulator (NVLE) is a general-purpose micro-computer and I/O interfacing unit that can perform the functions of various non-vital relay logic and digital logic systems, according to a custom-designed software program. This program uses a Boolean high-level language, taking inputs, performing logic and timing functions on those inputs and produces outputs. The program is conceptually similar to a system of interconnected relays. Typical applications include processing of central office controls and local indications at a CTC system field station, and processing of local indications for a local wayside control panel.

The basic hardware elements of GENISYS include a single microprocessor-based controller printed circuit board (PCB), a power supply converter PCB and a configuration of input and output PCBs determined by application. Three optional output PCBs are equipped with 16 single-pole relay outputs for various relay and/or control panel lamp driving applications. Two optional input PCBs are equipped with 16 optical-isolator ICs for interfacing of contact-generated input signals. Power supply converter options are tailored for dc voltages in the range of 9.5 to 35 Vdc and standard 120 Vac commercial power.

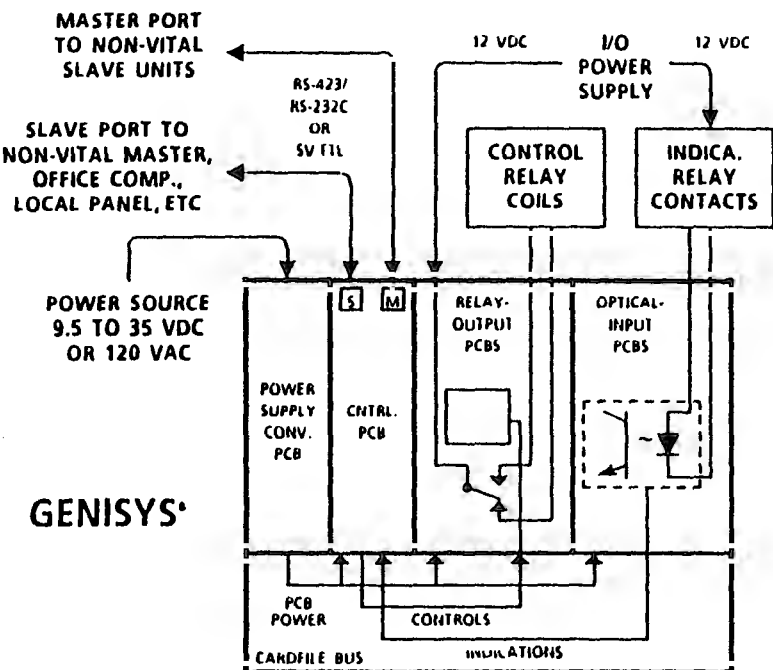
GENISYS may be interfaced directly with a controlling computer, and with digital carrier system modems. A Master/Slave protocol is used for communications between a computer and GENISYS unit(s) or between two or more units. Configurations may include (1) stand-alone units with no serial communications links and (2) Master/Slave systems with serial communications links. Master/Slave systems may, in turn, incorporate one or more Master GENISYS units. These configurations are shown in Figure 2-1.

Communications options include EIA RS-423 (RS-232C compatible) or TTL-compatible. Modems are required when more than one Slave unit is connected to a Master unit, or when direct-interface communications limits are exceeded. Each GENISYS unit may control a total of 256 inputs and outputs in any configuration, including those where inputs or outputs are absent. Up to 255 Slave units may be controlled by a Master unit, however the typical practical limit is in range of 40 to 50 units.

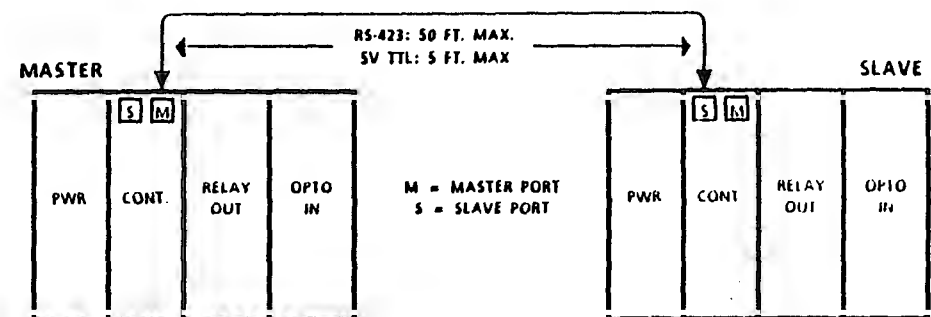
2.1.2 Application and Executive Software

The GENISYS system incorporates independent Application and Executive software. These are contained in separate EPROM chips on the Controller board.

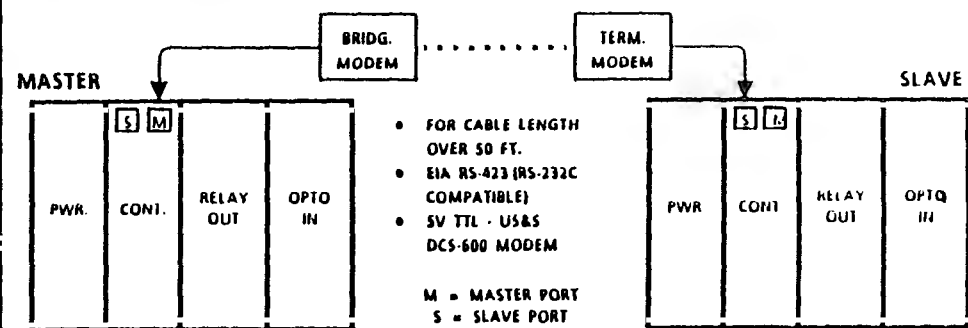
The Application software or logic is developed for the specific installation, either by US&S or the customer. The source program is written and "compiled" on a computer, using a language that enables the system logic to be expressed in terms familiar to the railroad engineer. The finished program is converted into a form that can be entered or "burned" into the EPROM chips.



SERIAL I/O - DIRECT INTERFACE

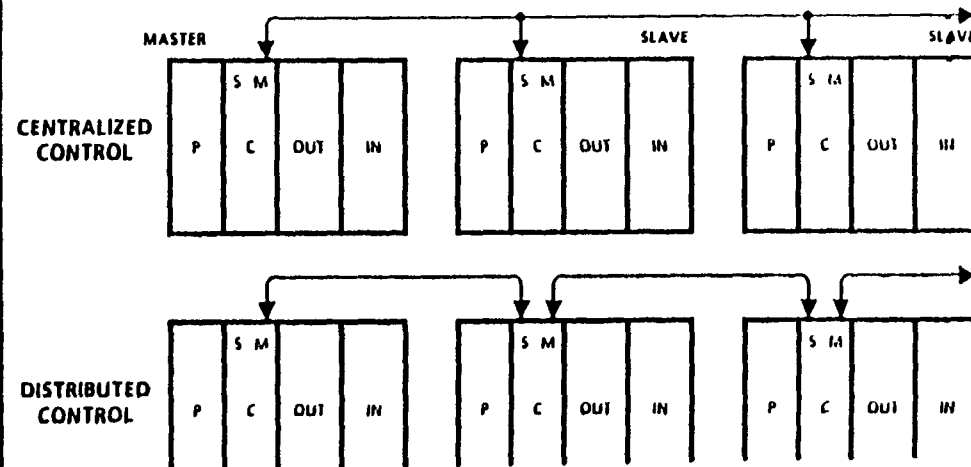


SERIAL I/O - MODEM INTERFACE



SERIAL I/O - SYSTEM CONFIGURATIONS

DESIGN MAX. = 255 SLAVES
PRACTICAL LIMIT = 45 TO 50 SLAVES



LOCAL I/O

TOTAL LOCAL I/O PER UNIT:

- 256 RELAY-OUTPUT AND/OR OPTICAL INPUT
- 256 RELAY-OUTPUT ONLY
- 256 OPTICAL-INPUT ONLY

RELAY-OUTPUT PCBs

- 16 PER PCB
- N451441-3601: EXTERNAL-STROBE OUTPUT
- N451441-4701: INTERNAL-STROBE OUTPUT
- N451441-7101: CONSTANT STATE AT POWER OFF

OPTICAL-INPUT PCBs:

- 16 PER PCB
- N451441-5802: 5 TO 32 VDC INPUTS
- N451441-7202: 5 TO 32 VDC INPUT (ASSURED 5 VDC THRESHOLD)

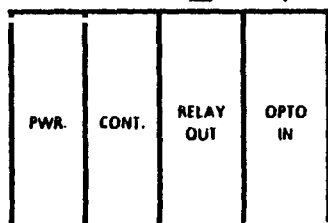


Figure 2-1. Basic GENISYS System

GENISYS may be programmed by using the GENISYS Development System (G.D.S). This system enables the user to design and test his own program, and then load it into the system hardware. The G.D.S. consists of a personal computer, an EPROM Programmer and the GENISYS software, which is contained on a single diskette. The G.D.S. can only be employed with the programming equipment supplied by US&S.

The Executive software is common to all GENISYS systems. This software performs input, internal and output logic operations defined in the application logic.

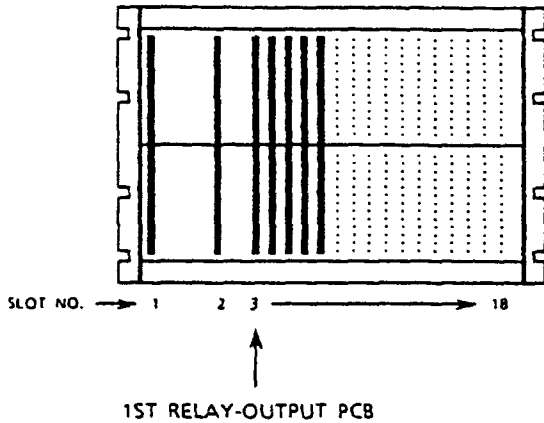
This manual edition covers all software versions up to and including Executive Software Version 11 and Application Compiler Software Version 3.0.

2.2 COMPONENTS

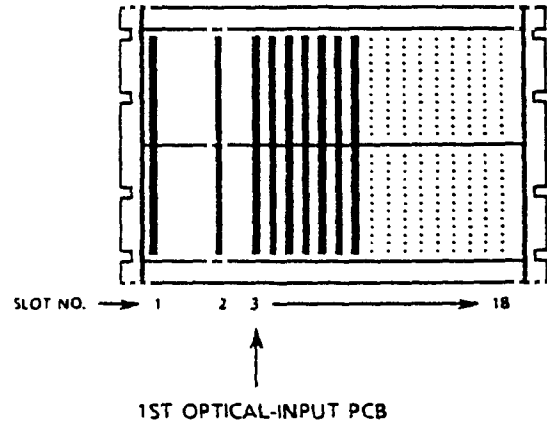
2.2.1 Cardfile (see page v and Figure 2-2)

The GENISYS Non-Vital Logic Emulator is housed in a standard 19 inch rack-mount cardfile. The cardfile always contains a power supply converter PCB in the far lefthand slot, a controller PCB in the second slot and between zero and 16 relay-output and/or optical-input PCBs in the remaining slots. When relay-output PCBs are present, they are always placed as a group to the left of any optical-input PCBs. Empty slots are allowed between relay-output and/or optical-input boards provided these slots are defined as "spares" in the application program. For example, in Figure 2-2, empty slots 7 and 8 ("spare") allow future expansion of outputs without having to reconfigure the inputs.

EXAMPLE: 5 RELAY OUTPUT PCBS ONLY



EXAMPLE: 7 OPTICAL-INPUT PCBS ONLY



GENISYS® PLUG-IN CIRCUIT BOARDS		
SLOT NO.	DESCRIPTION	PART NO.
1*	POWER SUPPLY (9.5 - 35 VDC)	N451441-7601
	POWER SUPPLY (120 VAC)	N451441-4601
2*	CONTROLLER	N451441-5602
3-18	RELAY-OUTPUT OPTIONS:	
	CONTROL DELIVERY	N451441-3601
	CONSTANT DELIVERY	N451441-7101
	CONTROL AND DELIVERY	N451441-4701
	OPTICAL-INPUT OPTIONS:	
	OPTO-INPUT (HIGH THRES.)	N451441-7202
	OPTO-INPUT (WIDE RANGE)	N451441-5802
*THESE PCBS ALWAYS IN THESE SLOTS		

EXAMPLE: 4 RELAY-OUTPUT AND 4 OPTICAL-INPUT PCBS

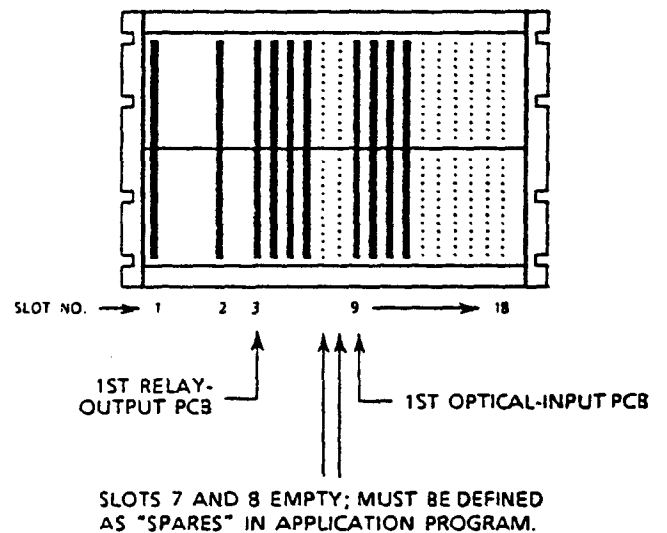


Figure 2-2. GENISYS Cardfile PCB Arrangements

2.2.2 PCBs

2.2.2.1 Controller N451441-5602

The Controller PCB performs all logical decisions and calculations for the GENISYS system, and serves as the remote communications interface for any external devices. Primary functions include management of local I/O (via card file interface boards) and remote I/O (via serial data line) according to the custom-design program, and execution of internal watchdog and testing routines.

2.2.2.2 Relay-Output PCBs

US&S provides three different relay-output PCBs for the output group of the GENISYS cardfile (see Figure 2-2). The cardfile need not contain any relay-output PCBs, or may contain these PCBs in all cardfile slots (maximum of 16). Applicable PCBs are as follows:

<u>PCB Name</u>	<u>Part Number</u>	<u>Operating Type</u>
Control Delivery	N451441-3601	Pulsed Delivery
Constant Delivery	N451441-7101	Stick Relays
Control and Delivery	N451441-4701	Internal Strobe

The -3601 PCB is the standard GENISYS output board. The remaining boards are used for special applications.

2.2.2.3 Optical-Input PCBs

US&S provides two different optical-input PCBs for the input group of the GENISYS cardfile (see Figure 2-2). Each is designed to handle different input voltage ranges and types. The cardfile need not contain any optical-input PCBs, or may contain these PCBs in all cardfile slots (maximum of 16). Applicable PCBs are as follows:

<u>Name</u>	<u>Part Number</u>
Indication-Opto	N451441-5802
Indication-Opto	N451441-7202

The -7202 PCB is the standard GENISYS input board.

2.2.2.4 Power Supply Converter PCBs

US&S provides two power supply converter PCBs for GENISYS which output operating power for other cardfile PCBs, and power for the carrier modem interface (+/- 12 Vdc) when required by application. These are as follows:

<u>Part Number</u>	<u>Input Voltage</u>
N451441-7601	9.5 to 35 Vdc
N451441-4601	120 Vac (nom.)

2.3 SPECIFICATIONS (Programming Related)

Total Bits:	1450 bits maximum (can be divided between local I/O, serial I/O and internal)
Local I/O Boards:	16 maximum per cardfile (any combination)
Local I/O Bits:	16 maximum per local I/O board
Master-to-Slave Communications:	255 Slave units maximum, communication from Master unit.
Slave-to-Master Communications:	1 Master unit maximum, communication from Slave unit.
Serial Addresses:	1 to 255 inclusive
Serial Baud Rates:	50, 75, 100, 134, 150, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, 9600
Active Timing Elements:	100 maximum active at any one time in application logic (more timers may be defined).
Logic Equations Triggered:	1000 maximum, one-queue option, 500 maximum each queue, two-queue option (refer to page 4-21).

SECTION III
GENERAL INFORMATION - MICROLOK PLUS

3.1 INTRODUCTION

3.1.1 Overall System (See Figure 3-1)

The MICROLOK PLUS Vital and Non-Vital Control Package is a multi-purpose, microprocessor-based device designed for use in both vital and/or non-vital railroad control systems. It is typically used for smaller applications, such as a single end-of-siding, that do not require the large input/output capabilities of separate vital and non-vital controllers. The device can be configured with a vital control section only, or with vital and non-vital control sections. (A non-vital-only configuration is also possible, but not typical.)

In a typical single end-of-siding application, both sections are utilized. The vital section controls the interlocking logic, manages switch machines, signals and track circuits in the control area; while the non-vital section provides an interface point for a local control panel, processes CTC office commands, and transmits indications from the vital section. Another typical end-of-siding configuration could consist of the vital section only, with code system inputs and outputs processed within the vital section (no local control panel).

The MICROLOK PLUS system is derived from the US&S MICROLOK Vital Interlocking Control System and the GENISYS Non-Vital Logic Emulator. It uses the same plug-in printed circuit boards, the same Executive software, and the same application logic compilers as the MICROLOK and GENISYS systems.

3.1.2 Non-Vital Section

The non-vital section of MICROLOK PLUS incorporates a Master port that enables the device to serve as the managing unit of a non-vital Master/Slave system. Up to 255 Slave units can be controlled from this port, although the practical upper limit is about 40 to 50 units. The Slave units for such a system might include additional MICROLOK PLUS or GENISYS units, or a combination of both. The non-vital Slave port enables the unit to function as a Slave to another Master unit. The Master unit can include another MICROLOK PLUS unit (non-vital section), a GENISYS system or an office computer.

When the application requires a serial link between the non-vital and vital sections of the MICROLOK PLUS unit, the non-vital Master port is used for this purpose, while the Slave is used for external communications.

Non-vital serial link communications are formatted to EIA RS-423 standards, and derated to operate under the RS-232C standards. This enables the MICROLOK PLUS non-vital serial ports to be interfaced to an EIA-compatible modem for remote communications.

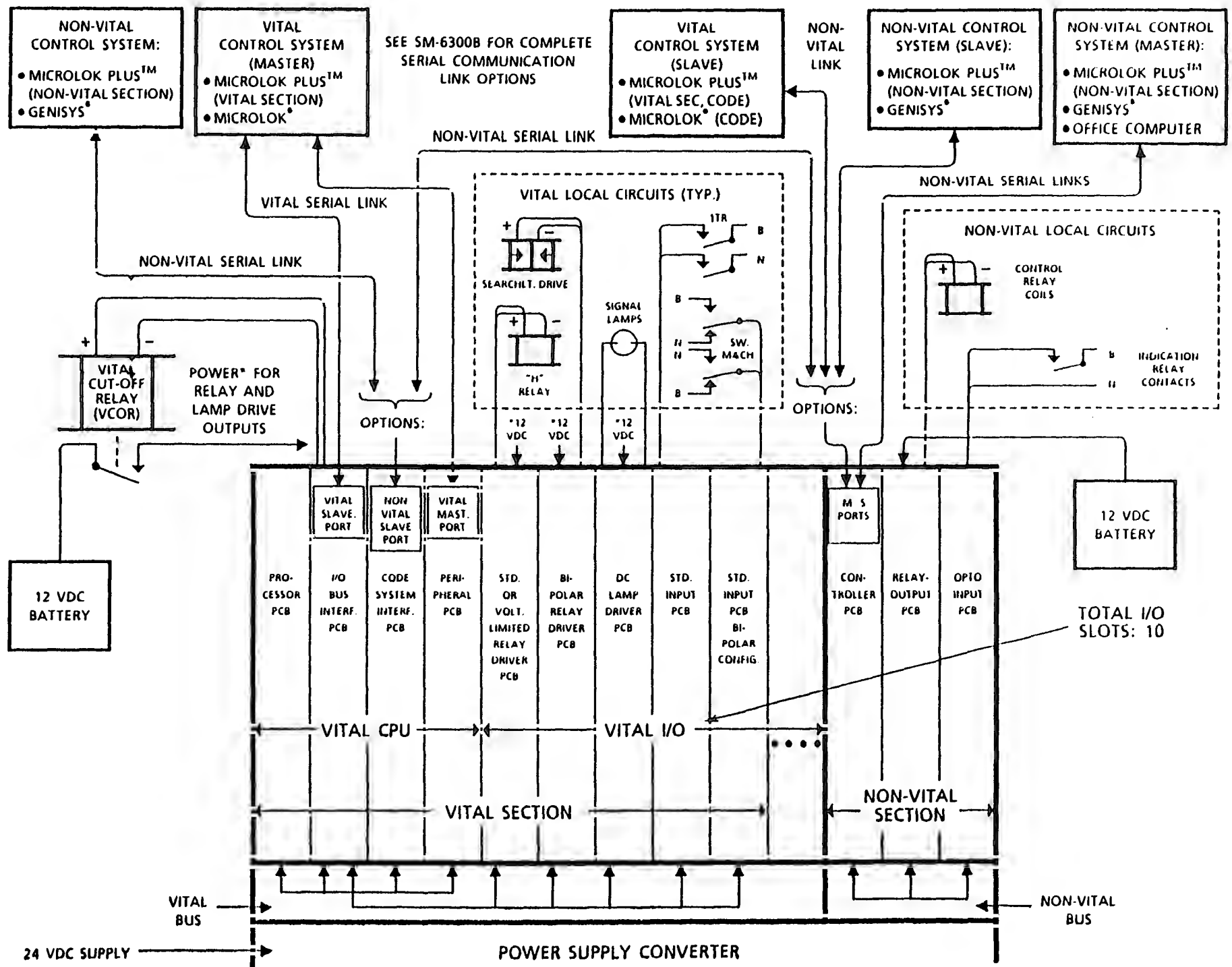


Figure 3-1. Basic MICROLOK PLUS System

6300A, p. 3-2

3.1.3 Application and Executive Software (See Figure 3-2)

Each section (vital and non-vital) of the MICROLOK PLUS unit incorporates its own, independent Application and Executive software. These are contained in EPROM chips on the logic boards of the respective sections.

The Application software or logic in each section is developed for the specific installation, either by US&S or the customer. The source program is written and "compiled" on a computer, using a language that enables the system logic to be expressed in terms familiar to the railroad engineer. The finished program is converted into a form that can be entered or "burned" into the EPROM chips.

The MICROLOK PLUS non-vital application logic can be developed using the optional GENISYS Development System (G.D.S.). This system enables the user to design and test his own program, and then load it into the system hardware. The G.D.S. consists of a personal computer, an EPROM Programmer and the GENISYS software, which is contained on a single diskette. The G.D.S. can only be employed with the programming equipment supplied by US&S.

The Executive software of the vital and non-vital sections of MICROLOK PLUS are common to all units. The non-vital section Executive software performs input, internal and output logic operations defined in the non-vital application logic.

This manual edition covers all software versions up to and including Executive Software Version 11 and Application Compiler Software Version 3.0.

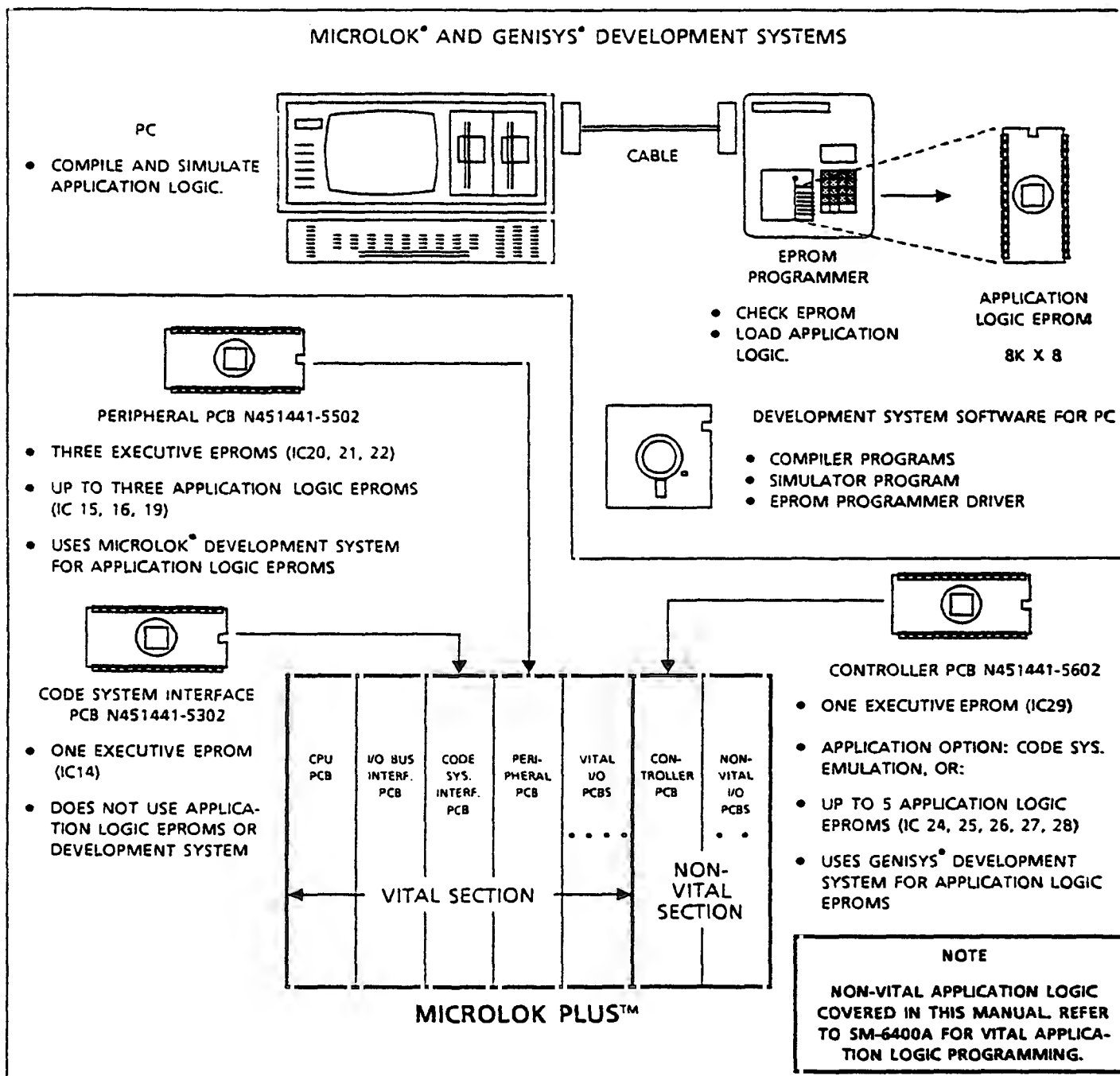


Figure 3-2. MICROLOK PLUS Application and Executive Software

3.2 COMPONENTS

3.2.1 Cardfile (see page v and Figure 3-3)

The MICROLOK PLUS package is housed in a printed circuit board cardfile designed for mounting in a standard 19 inch equipment rack. Vital section boards are placed in the ten left-most slots, while non-vital boards are placed in the three right-most slots. The non-vital section is controlled by a single Controller PCB. Two slots are provided to the right of the Controller PCB for local interfacing. When the application requires only one relay-output or optical-input board, this board is always installed in slot P. When the application requires one relay-output and optical-input board, the relay-output board is always installed in slot P and the optical-input board is always installed in slot Q.

3.2.2 PCBs

3.2.2.1 Controller N451441-5602

The Controller PCB performs all logical decisions and calculations for the MICROLOK PLUS non-vital section, and serves as the remote communications interface for any external devices. Primary functions include management of local I/O (via card file interface boards) and remote I/O (via serial data line) according to the custom-design program, and execution of internal watchdog and testing routines.

3.2.2.2 Relay-Output PCBs

US&S provides three different relay-output PCBs for the MICROLOK PLUS non-vital section outputs. Applicable PCBs are as follows:

<u>PCB Name</u>	<u>Part Number</u>	<u>Operating Type</u>
Control Delivery	N451441-3601	Pulsed Delivery
Constant Delivery	N451441-7101	Stick Relays
Control and Delivery	N451441-4701	Internal Strobe

The -3601 PCB is the standard MICROLOK PLUS non-vital output board. The remaining boards are used for special applications.

3.2.2.3 Optical-Input PCBs

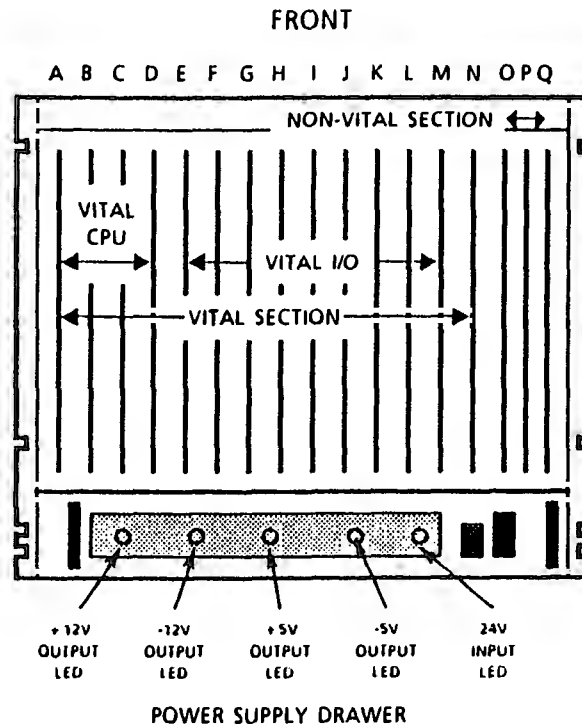
US&S provides two different optical-input PCBs for the MICROLOK PLUS non-vital section inputs. Each is designed to handle different input voltage ranges and types. Applicable PCBs are as follows:

<u>Name</u>	<u>Part Number</u>
Indication-Opto	N451441-5802
Indication-Opto	N451441-7202

The -7202 PCB is the standard MICROLOK PLUS non-vital input board.

Figure 3-3. MICROLOK PLUS Cardfile PCB Arrangement

VITAL SECTION PCBS - CPU		
SLOT	PART NO.	DESCRIPTION
A	N451441-5701	PROCESSOR
B	N451441-6001	I/O BUS INTERFACE
C	N451441-5302	CODE SYSTEM INTERFACE
D	N451441-5502	PERIPHERAL
INSTALLATION PROCESSOR SLOT A, ALL VITAL APPLICATIONS I/O BUS INTERFACE SLOT B, ALL VITAL APPLICATIONS CODE SYSTEM INTERFACE SLOT C, WHEN REQ'D BY APPLICATION PERIPHERAL SLOT D, ALL VITAL APPLICATIONS		
VITAL SECTION PCBS - I/O		
SLOT	PART NO.	DESCRIPTION
E - N	N451441-8601	STANDARD RELAY DRIVER
	N451441-8501	VOLT-LIMIT RELAY DRIVER
	N451441-8701	BI-POLAR RELAY DRIVER
	N451441-6702	DC LAMP DRIVER, 18 W LAMP
	N451441-6703	DC LAMP DRIVER, 25 W LAMP
	N451441-7301	DC LAMP DRIVER, 36 W LAMP
	N451441-8802	STANDARD INPUT, 12 V NOM
	N451441-8803	STANDARD INPUT, 24 V NOM.
INSTALLATION: SEE EXAMPLES BELOW		

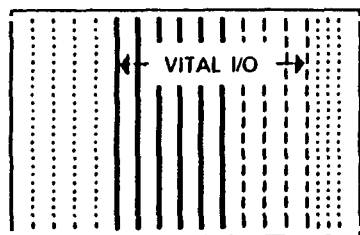


NON-VITAL SECTION PCB - LOGIC		
SLOT	PART NO.	DESCRIPTION
O	N451441-5602	CONTROLLER
INSTALLATION SLOT O, ALL NON-VITAL APPLICATIONS		
NON-VITAL SECTION PCBS - I/O		
SLOT	PART NO.	DESCRIPTION
P, Q	N451441-3601	CONTROL DELIVERY
	N451441-7101	CONSTANT DELIVERY
	N451441-4701	CONTROL & DELIVERY
	N451441-7202	OPTO-INPUT (HIGH THRESHOLD)
	N451441-5802	OPTO-INPUT (WIDE RANGE)
INSTALLATION ONE RELAY-OUTPUT: SLOT P ONE OPTO-INPUT: SLOT P ONE RELAY-OUTPUT AND ONE OPTO-INPUT. RELAY IN SLOT P, OPTO IN SLOT Q		

44-WAY PCB EDGE CONNECTORS (VITAL I/O AND ALL NON-VITAL PCBS) VITAL MASTER PORT 37-PIN "D" CODE SYSTEM PORT 25-PIN "D" VITAL SLAVE PORT 37-PIN "D"

VITAL I/O INSTALLATION EXAMPLES

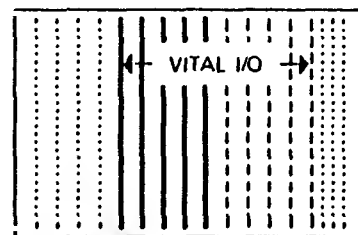
EXAMPLE: 6 OUTPUT PCBS ONLY



1ST OUTPUT PCB IN SLOT E

NO EMPTY SLOTS BETWEEN PCBs

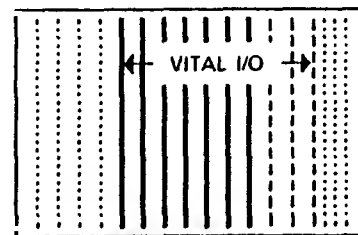
EXAMPLE: 5 INPUT PCBS ONLY



1ST INPUT PCB IN SLOT E

NO EMPTY SLOTS BETWEEN PCBs

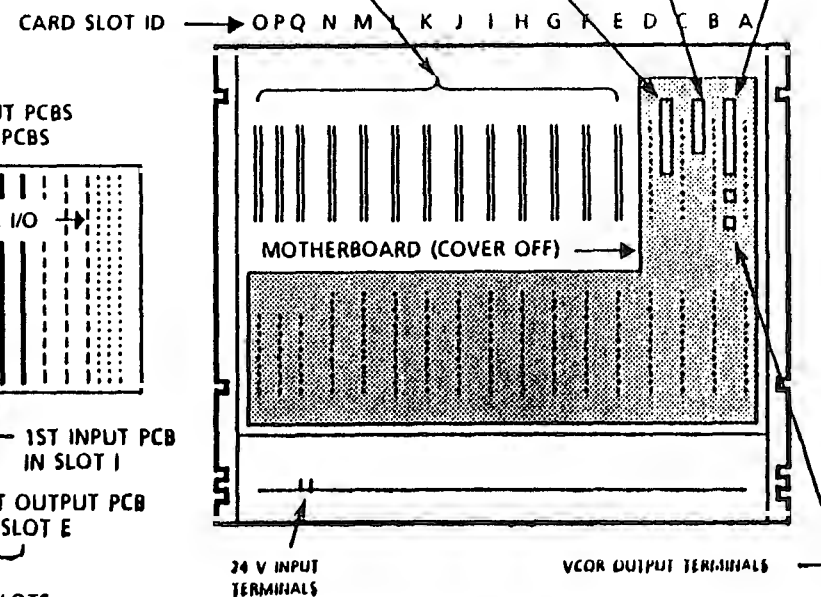
EXAMPLE: 4 OUTPUT PCBS 3 INPUT PCBS



1ST INPUT PCB IN SLOT I

1ST OUTPUT PCB IN SLOT E

NO EMPTY SLOTS



3.3 NON-VITAL SECTION SPECIFICATIONS (Programming Related)

Total Bits:	1450 bits maximum (can be divided between local I/O, serial I/O and internal)
Local I/O Boards:	2 maximum per cardfile
Local I/O Bits:	16 maximum per local I/O board
Master-to-Slave Communications:	255 Slave units maximum, communication from Master unit.
Slave-to-Master Communications:	1 Master unit maximum, communication from Slave unit.
Serial Addresses:	1 to 255 inclusive
Serial Baud Rates:	50, 75, 100, 134, 150, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, 9600
Active Timing Elements:	100 maximum active at any one time in application logic (more timers may be defined).
Logic Equations Triggered:	1000 maximum, one-queue option, 500 maximum each queue, two-queue option (refer to page 4-21).

SECTION IV PROGRAMMING PROCEDURES - GENISYS AND MICROLOK PLUS

4.1 GENERAL

In the non-vital application program, various bits (input, output, internal etc.) and logic procedures are defined in a text data file on a computer. Input and output statements are described by location. They can be local I/O connected directly to that unit, or remote I/O on either of two serial lines (Master, Slave). Timing values indicate the set (pick-up) and/or clear (drop-away) delays of the bits. Boolean statements are used to describe the system logic. The completed "written" program is referred to as the source program. It is processed by the compiler and converted into data tables. In turn, these tables are "burned" into one or several EPROMs; the EPROM(s) are then installed into the Controller PCB. Sections 4.3 to 4.8 describe how these basic programming operations are handled with the US&S GENISYS Development System (G.D.S.).

4.2 PROGRAMMING LANGUAGE

4.2.1 Terms

4.2.1.1 Character Set

The Character Set consists of the full ASCII character set, as defined for the user's computer. These are listed in Table 4-1. Only certain characters may be used to make up user-defined symbols (refer to section 4.2.1.3). Although all letters are acceptable, all lower case letters (a-z) are converted to upper case (A-Z) by the system. For example, "Stick" is read the same as "STICK". Lower case is only used for readability.

Table 4-1. Character Set

<u>FOR USER-DEFINED SYMBOLS</u>	<u>SPECIAL CHARACTERS</u>	<u>SPECIAL CHARACTERS</u>
Upper Letters (A-Z)	Colon :	Backslash \
Lower Letters (a-z)	Semicolon ;	Percent Sign %
Numerals (0-9)	Comma ,	At Sign @
Period .	Equal Sign =	Plus Sign +
Dollar Sign \$	Open Parens. (Asterisk *
Underscore _	Close Parens.)	Tilda ~

4.2.1.2 Reserved Words

A Reserved Word has a predefined meaning to the compiler. The 25 Reserved Words are listed in Table 4-2:

Table 4-2. GENISYS Reserved Words

PROGRAM	OUTPUT	SET	AND	MSEC
INTERFACE	INPUT	CLEAR	OR	SEC
LOCAL	WORD	BEGIN	NOT	MIN
MASTER	VAR	END	XOR	ADDRESS
SLAVE	TIMER	ASSIGN	TO	SPARE

4.2.1.3 User-Defined Symbols

User-defined symbols are used to create relay names in the source program. These symbols must contain characters from the first part of Table 4-1, and cannot consist of all numbers. A maximum of 12 characters may be used to create symbol names.

Examples of legal symbols in G.D.S. Versions 1.01 and higher include:

relay.123	DOG
.INPUT RELAY	1TK
INSTRK\$1N	

Illegal examples of the above symbols in G.D.S. Versions 1.01 and higher include:

.INPUT.RELAY.1	(Exceeds 12 character maximum)
RELAY#1	(Contains an illegal character)
123	(all numbers)

Examples of legal symbols in G.D.S. Version 1.00 include:

relay.123	INSTRK\$1N
.1TK	DOG
.INPUT_RELAY	

Illegal examples of the above symbols in G.D.S. Version 1.00 include:

1TK	(Begins with number)
.INPUT.RELAY.1	(Exceeds 12 character maximum)
RELAY#1	(Contains an illegal character)

4.2.1.4 Delimiters

Delimiters separate individual words. Delimiters in the non-vital program language are listed in Table 4-3.

Table 4-3. Delimiters

space	semicolon	(;)	open parenthesis	((
tab	equal sign	(=)	close parenthesis	())
colon	comma	(,)	carriage return	(CR)
backslash	percent	(%)	tilda	(~)
"at"	plus sign	(+)	Asterisk	(*)

Every distinct word or token in the source program must be separated by one of the above delimiters. Extra space and tab delimiters may be inserted anywhere in the program; they have no effect on the meaning of the program.

4.2.2 Formats

4.2.2.1 General Arrangement of Statements

Source program statements may begin anywhere on a line with tabs. Non-significant spaces are ignored by the compiler. If a statement is too long to fit on one line, it may be continued to any number of following lines as required. The maximum allowable line length is 100 characters. If this is exceeded, an error message will be generated. Although the non-vital compiler uses a free format, statements should be arranged for easy reading.

4.2.2.2 Non-Program Comments

Miscellaneous comments may be inserted in the source program to aid the user in charting and reviewing the program. To distinguish a non-program comment from program statements, begin the statement with a (%) and end with a backslash (\). For example:

```
% THIS IS AN EXAMPLE OF A LEGAL GENISYS OR MICROLOK PLUS COMMENT\
```

When the compiler encounters the percent (%) sign, characters are ignored until it reaches a backslash (\). Switches are the exception (refer to section 4.2.2.3). Note that comments may begin anywhere (including the middle of a statement) and span any number of lines in a source program. However, they cannot begin in the middle of a word (ASSI%GN is illegal). Hence, another example of a correct comment is as follows:

```
%  
  THIS IS AN EXAMPLE OF  
  A LEGAL GENISYS OR MICROLOK PLUS  
  COMMENT  
\  
The closing backslash (\) must be inserted, otherwise the compiler ignores all other characters until a backslash is found.
```

Compiler Version 3.0 and Higher - Note the in the program example on page 4-30 that an exclamation point appears before each comment. This character only appears if the comment is the first non-blank item on the line or spans multiple lines and is automatically inserted after the line number to help distinguish the comment from other parts of the program. Thus, if the closing backslash is accidentally omitted from the comment, the exclamation point will appear after every line number to indicate that the compiler regards all subsequent lines as comments, rather than other types of program statements.

4.2.2.3 Compiler Switches

Compiler switches are used in the source program to select various options such as baud rates. (Refer to Section 8.4 for hardware switch options.)

- Compiler switches begin with a percent sign (%) (like non-program comment)
- To distinguish it from a comment, a dollar sign (\$) must be placed immediately after the percent sign.
- After the dollar sign is a single letter representing the switch name.
- Next is a character(s) representing the value of the switch.

All characters after the value character(s) are ignored by the compiler until the next backslash symbol (\).

<u>Example</u>	<u>Comment</u>
%\$S3\	Sets switch "S" to value 3.
%\$D- THIS IS A COMMENT FOLLOWING THE SWITCH\	Sets switch "D" to off. Remainder is a comment.

Compiler switches are as follows:

%\$Sn\ Set the Baud Rate on the Slave Port (where n = 1 through Hexadecimal E

This value is defined on the hardware switch or in the source program, and is a single character. If no Slave baud rate switch is present in the source program, the rate set on switch SW1 will be used. Software values for this function are listed in Table 4-4 on page 4-7.

NOTE

In Executive software revisions 0 through 10 of the following switch (%\$Mn), a setting of n = 0 would generate an error. The revision 11 function is as follows:

%\$Mn\ Set the Baud Rate on Master Port (where n = 0 through Hexadecimal E)

This value is set only in the source program, and is a single character. If no Master baud rate switch is present in the source program, the default rate is 1200 BPS. If "0" is specified, the compiler defaults to the setting on rotary switch SW1 on the Controller PCB. Software values and associated baud rates are listed in Table 4-5 on page 4-7.

NOTE

The following compiler switch (`%%Pxxxx`) applies to GENISYS Development System Versions 1.02 and higher and Executive PROM IC29 Revision 7 and higher. This switch is not available with earlier versions.

`%%Pxxxx\` Specify the Master No-Response Time-Out (where x = 100 through 9999)

This value is set only in the source program. It specifies the number of milliseconds the Master will wait for the Slave to begin responding. If the Slave has not responded within this time, the message is considered bad and the Master will continue with the polling cycle. The value entered is rounded upward to the next multiple of 100. For example, `%%P735\` (polling time-out = 735 milliseconds) would instruct the executive software to wait 800 milliseconds for the Slave to respond. If this switch is not specified, the default is 1 second.

`%%$On\` Set the Control Delivery time (where n = 1 through Hexadecimal F)

This value is the duration of the Controller PCB delivery pulse to the relay output PCB(s). It is defined on the hardware switch or in the source program, and is a single character. If no `%%$O` switch is present in the source program, the Control Delivery time set on switch SW2 will be used. Software values are listed in Table 4-6 on page 4-7.

NOTE

The following compiler switch (`%%Cv\`) applies to GENISYS Development System Versions 1.01 and higher. In version 1.00, this switch is listed as Security On/Off. (There is no operational difference between the two Versions.)

`%%Cv\` Security On/Automatic (where v = + or -)

Slave unit polling from a Master port in a GENISYS or MICROLOK PLUS system can occur with or without CRC security (refer to protocols in Section VII). The CRC security is either "+" (send poll with CRC security) or "-" (automatic on/off CRC security for poll). When the "+" value is selected, the Master unit will always send the secure form of the polling command. If the "-" option is selected, the security is automatically included when the Master determines that the line is sufficiently noisy to warrant it. The default for this option is `%%C-\` (automatic CRC inclusion).

`%%Vv\` Validate Value (where v = + or -)

This switch is used to inform the Executive software whether or not to wait for validation before an output is performed. Refer to sections 4.2.5.4 and 5.3 for detailed descriptions of this switch. The value (v) of this switch is either plus or minus (+/-). Plus corresponds to option "on" and minus to option "off". The default for this switch is `%%V+\` (validation on).

%\$Dv\ Debug (where v = + or -)

G.D.S. Versions 1.01 and Higher:

This switch informs the compiler that the simulator will be used to debug the program. When this switch is turned on (%\$D+ \), the compiler retains, in a separate file, relay names assigned in the application program so that these names may be used when simulating the application logic. Programs to be run on the simulator must have the D+ switch. The program does not have to be recompiled with %\$D- \ to permit programming of PROMs. The default for this switch is %\$D- \ .

G.D.S. Version 1.00:

(Same function). After debugging, the program must be recompiled with %\$D- \ to permit programming of EPROMs. The default for this switch is %\$D- \ .

%\$Qv\ Queuing (where v = + or -)

This switch sets the number of logic queues to be used. One queue or two queues may be selected. Refer to section 4.2.5.2 for a detailed description of this switch.

%\$Bv\ Symbol Table Listing (where v = + or -)

This switch inhibits the symbol table from being placed in the list file. When set to %\$B- \ , this switch turns off the symbol table at the bottom of the compiler listing. If the validation switch (refer to previous page) is enabled, the program should be compiled with the symbol table enabled before the PROM is programmed. This will insure that no unassigned outputs exist. If any exist, they should be assigned values and the program recompiled to avoid validation errors. The default for this switch is %\$B+ \ .

NOTE

The following compiler switch (%\$E \) applies to
GENISYS Development System Versions 3.0 and higher.
This switch is not available with earlier versions.

%\$E\ Page Generator

When the compiler encounters the %\$E \ switch, the next source line is placed at the top of a new page in the compiler listing.

Table 4-4. Software Switch Settings for Slave Port Baud Rate

Switch Value "\$S"	Baud Rate	Switch Value "\$S"	Baud Rate
1	50	9	1800
2	75	A	2400
3	110	B	3600
4	134	C	4800
5	150	D	7200
6	300	E	9600
7	600		
8	1200		

Table 4-5. Software Switch Settings for Master Port Baud Rate

Switch Value "\$M"	Baud Rate	Switch Value "\$M"	Baud Rate	Switch Value "\$M"	Baud Rate
0	Hardware*	5	150	A	2400
1	50	6	300	B	3600
2	75	7	600	C	4800
3	110	8	1200	D	7200
4	134	9	1800	E	9600

*Defaults to baud rate set on Controller PCB rotary switch SW1. Only applicable to Executive software revisions 11.0 and higher.

Table 4-6. Software Switch Settings for Control Delivery Time

Software Switch "\$S0" Setting	Software Switch Control Del. Time	"\$S0" Setting	Control Del. Time
1	10 msec.	9	4 sec.
2	30 msec.	A	8 sec.
3	70 msec.	B	16 sec.
4	130 msec.	C	30 sec.
5	250 msec.	D	1 min.
6	500 msec.	E	2 min.
7	1 sec.	F	4 min.
8	2 sec.		

4.2.3 Program Examples

4.2.3.1 Local Input/Output

The following sample program shows the basic local input/output and logic features of the non-vital program language:

```
PROGRAM NBR1;
INTERFACE
  LOCAL
    OUTPUT WORD:
      OUT.1, OUT.2, OUT.3, OUT.4, OUT.5;
    INPUT WORD:
      IN.A, IN.B, IN.C;
BEGIN
  ASSIGN IN.A AND IN.B          TO OUT.1;
  ASSIGN IN.A XOR IN.B          TO OUT.2;
  ASSIGN IN.A OR IN.B           TO OUT.3;
  ASSIGN NOT IN.A               TO OUT.4;
  ASSIGN IN.C                   TO OUT.5;
END
```

The major sections of the program (PROGRAM, INTERFACE, LOCAL, OUTPUT WORD etc.) are always placed in the order shown. These are discussed further in section 4.2.4.

The INTERFACE section defines the local inputs and outputs, corresponding to the relay-output and optical input boards, and individual bits on those boards, in the cardfile. This sample program requires one LOCAL relay-output board and one LOCAL optical-input board. The output board and input board must be installed as follows:

GENISYS - Output board in slot #1 (J3 of the cardfile) and input board in slot #2 (J4 of the cardfile).

MICROLOK PLUS - Output board in non-vital slot #1 (P of the cardfile) and input board in non-vital slot #2 (Q of the cardfile).

Five output bits (OUT.1, OUT.2, OUT.3, OUT.4, OUT.5) are defined on the output board. OUT.1 is delivered on the first output (bit 0). OUT.2 through OUT.5 are delivered through bits 1 through 4, respectively. Three inputs are similarly defined for inputs 0 through 2.

The number and order of the I/O "boards" in the program text must match the actual hardware configuration. Output board definitions must be specified before input board definitions in the program. This is in accordance with the left-to-right order of I/O boards in the cardfile (see Figures 2-2 and 3-3). Less than 16 bits may be defined on an input or output board. Unused input bits are ignored. Unused output bits are always output as zero.

NOTE

The GENISYS Development System compiler permits up to 16 I/O PCBs to be defined. However, the MICROLOK PLUS non-vital section only allows two I/O PCBs. When writing the application program for the MICROLOK PLUS non-vital section, make certain not to specify more than two I/O boards. No error message will be generated if more than two boards are specified.

The actual system logic is defined with ASSIGN statements. These statements define the interconnecting logic of the inputs and outputs. The order of the ASSIGN statements will usually have no effect on the logic (refer also to section 4.2.5.2). In this example, OUT.1 is the logical "AND" of the two inputs IN.A and IN.B. Similarly, OUT.2 is the "EXCLUSIVE OR" and OUT.3 is the "OR" of the two inputs. OUT.4 is the logical "NOT" of IN.A, and OUT.5 directly follows IN.C.

4.2.3.2 Internal Relays and Stick Logic

The following program shows the handling of internal relays and stick logic:

```
PROGRAM NBR2;
INTERFACE
  LOCAL
    OUTPUT WORD:
      OUT.1, OUT.2, OUT.3, OUT.4, OUT.5, OUT.6;
    INPUT WORD:
      IN.A, IN.B, IN.C, IN.D;

  VAR
    STICK;

  BEGIN
    ASSIGN IN.A AND IN.B          TO OUT.1;
    ASSIGN IN.A XOR IN.B          TO OUT.2;
    ASSIGN IN.A OR IN.B           TO OUT.3;
    ASSIGN NOT IN.A               TO OUT.4;
    ASSIGN IN.C                   TO OUT.5;
    ASSIGN IN.C OR (STICK AND NOT IN.D) TO STICK;
    ASSIGN STICK                  TO OUT.6;

  END
```

This program is similar to the program "NBR1" in section 4.2.3.1, however OUT.6 is added as the 6th output on the relay-output board. Also, input IN.D, is added as the 4th input bit. A new section VAR is added to define an "internal bit". An internal bit is neither input nor output, it is only processed internally. All internal VAR bits are initially deenergized. An example is a stick relay. Note the additional ASSIGN statement:

```
ASSIGN IN.C OR (STICK AND NOT IN.D)          TO STICK;
```

When input IN.C is energized, STICK becomes energized. With the following direct assignment of STICK to OUT.6, both OUT.6 and STICK will remain energized, even if IN.C is deenergized. The internal STICK and OUT.6 will remain energized until IN.D is energized. This results in the clearing of the stick circuit. Note, however, that if IN.C is still energized, STICK and OUT.6 will remain energized.

4.2.3.3 Timing Relays

The sample program below shows the handling of timing relays. This program makes use of one timer relay. Continuing from the example in section 4.2.3.2, two more defined output bits, OUT.7 and OUT.8 are added. In the VAR section, another internal relay ("T1") is defined. Relays with timing characteristics are defined in the TIMER section, always after the VAR section. Every bit name specified in a TIMER statement must be previously defined as an output or internal bit. Input bits may not have timing characteristics.

```

PROGRAM NBR3;
INTERFACE
  LOCAL
    OUTPUT WORD:
      OUT.1, OUT.2, OUT.3, OUT.4, OUT.5, OUT.6 OUT.7, OUT.8;
    INPUT WORD:
      IN.A, IN.B, IN.C, IN.D;
  VAR
    STICK, T1;
  TIMER
    T1:      SET = 1:SEC      CLEAR = 1:SEC
  BEGIN
    ASSIGN IN.A AND IN.B      TO OUT.1;
    ASSIGN IN.A XOR IN.B      TO OUT.2;
    ASSIGN IN.A OR IN.B       TO OUT.3;
    ASSIGN NOT IN.A           TO OUT.4;
    ASSIGN IN.C               TO OUT.5;
    ASSIGN IN.C OR (STICK AND NOT IN.D) TO STICK;
    ASSIGN STICK              TO OUT.6;
    ASSIGN NOT T1             TO T1;
    ASSIGN T1                 TO OUT.7;
    ASSIGN NOT T1             TO OUT.8;
  END

```

Note the ASSIGN statement:

```
    ASSIGN NOT T1                TO T1;
```

The statement will take the current value of bit "T1" (initially 0), perform the logical "NOT" operation, and attempt to assign a value of "1" to T1. However, because T1 is defined to have a SET or pick-up delay of 1 second, the actual value will remain at 0 for one second. When time has elapsed and the bit becomes a "1", the assignment statement will execute again, causing the value of "0" to be assigned to "T1" with a one second delay (as specified by the CLEAR parameter of the timer statement). Thus, the internal bit, "T1", will toggle at a one second rate. Note the final two assignment statements added to this program:

```
    ASSIGN T1                    TO OUT 7;  
    ASSIGN NOT T1                TO OUT 8;
```

Outputs OUT.7 and OUT.8 will alternately flash at a one second rate (OUT.7 on when T1 is "1" and OUT.8 on when T1 is "0").

4.2.3.4 Master/Slave Communications

The following program shows the handling of Master/Slave serial communications for the Master unit:

```
PROGRAM NBR4M;  
INTERFACE  
  LOCAL  
    INPUT WORD:  
      IN.A, IN.B, IN.C;  
  MASTER  
    ADDRESS:2  
    OUTPUT:  
      M.OUT.1, M.OUT.2, M.OUT.3, M.OUT.4, M.OUT.5;  
BEGIN  
  ASSIGN IN.A AND IN.B      TO M.OUT.1;  
  ASSIGN IN.A XOR IN.B      TO M.OUT.2;  
  ASSIGN IN.A OR IN.B       TO M.OUT.3;  
  ASSIGN NOT IN.A           TO M.OUT.4;  
  ASSIGN IN.C               TO M.OUT.5;  
END
```

This sample shows Master/Slave serial communications for the Slave unit:

```
PROGRAM NBR45;
INTERFACE
  LOCAL
    OUTPUT WORD:
      OUT.1, OUT.2, OUT.3;
  SLAVE
    ADDRESS:2
    INPUT:
      M.IN.1, M.IN.2, M.IN.3, M.IN.4, M.IN.5;
BEGIN
  ASSIGN M.IN.1 OR M.IN.5      TO OUT.1;
  ASSIGN M.IN.4 XOR NOT M.IN.2 TO OUT.2;
  ASSIGN M.IN.3 AND M.IN.4     TO OUT.3;
END
```

The above programs allow communication between a Master and one Slave unit. The Slave unit responds to address 2. The Master unit inputs three bits of information, performs logic functions and sends five new output bits (M.OUT.1 through M.OUT.5) to the Slave unit. The Slave receives these bits (M.IN.1 through M.IN.5), performs additional logic and outputs the new computed values to the relay-output PCB in the cardfile.

For a complete explanation of Master/Slave communications, refer to section 4.2.4.2, part c.

4.2.4 Detailed Statement Descriptions

4.2.4.1 "PROGRAM" Statement

The first statement in the program must be a PROGRAM statement. This statement gives a name to the program for documentation purposes.

FORMAT:

```
PROGRAM <id> ;
```

The identifier entered here will be printed in the listing at the top of the symbol table. It is written with user-defined symbol (refer to section 4.2.1.3). Comments may be placed before a program statement.

4.2.4.2 "INTERFACE" Section

a. General

The INTERFACE section defines the various output and input specifications of the system. There are three sub-sections in the INTERFACE section: LOCAL I/O, MASTER I/O port and SLAVE I/O port. These correspond to the three input/output interfaces on the controller board. At least one of the three I/O interfaces must be defined. Each begins with a name (LOCAL, MASTER, or SLAVE) that designates the type of interface. The LOCAL I/O (if any) is defined first, followed by MASTER (if any) and then SLAVE (if any). Within each of these, outputs (if any) are defined first, followed by inputs (if any). Either outputs or inputs may be blank, but not both at the same time. The standard format is shown at the top of the next page.

b. LOCAL I/O

The LOCAL I/O subsection defines the names of the bits that are input and output on the cardfile I/O boards. (If there is no local I/O, this subsection may be omitted.)

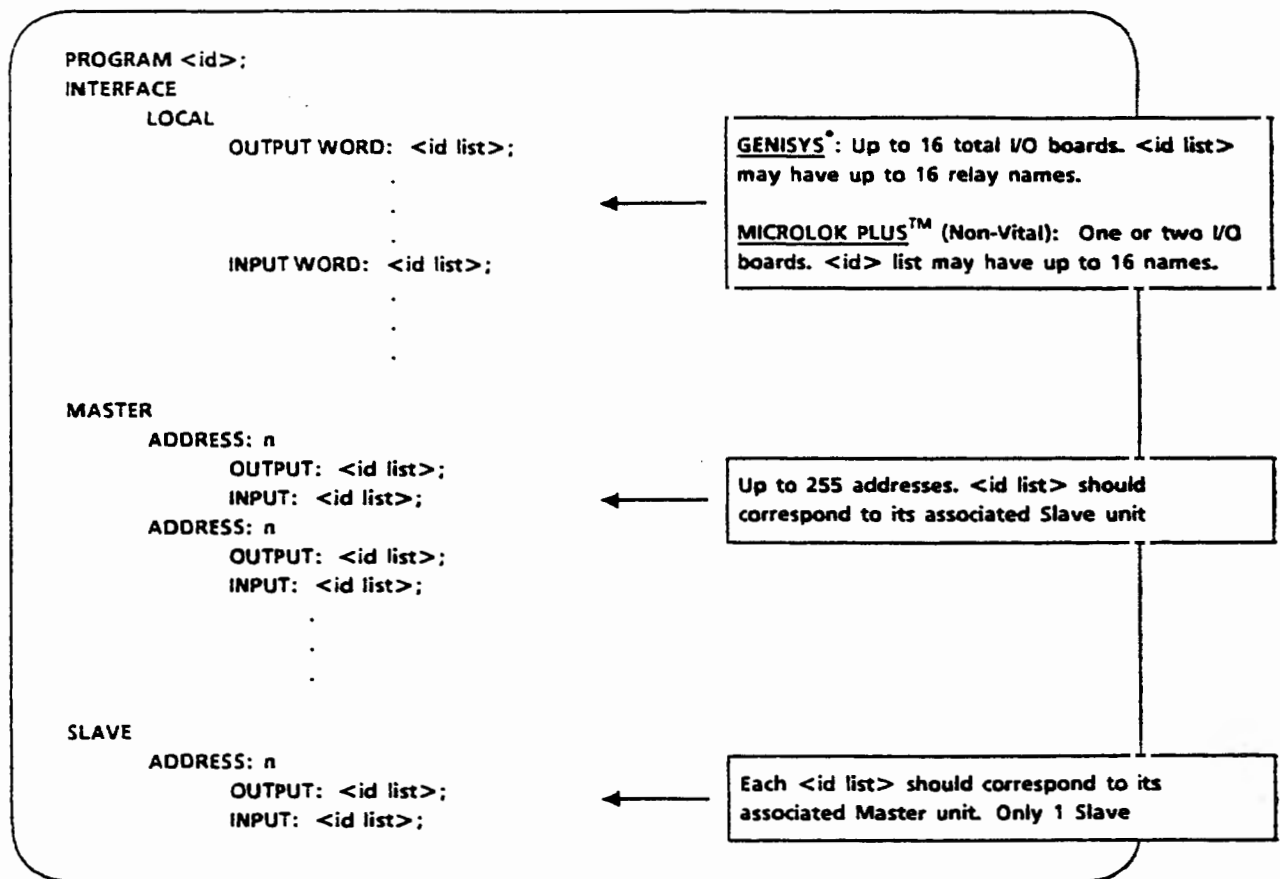
GENISYS - Up to 16 words may be defined, with 1 to 16 bit names on each word.

MICROLOK PLUS - One or two words may be defined, with 1 to 16 bit names on each word.

Each input or output WORD corresponds to a single I/O board. The first symbol defined on each word corresponds to the first input or output on the corresponding board. The second symbol corresponds to the second input or output, and so on. On interfaces where there is an unused bit between two active bits, the symbol SPARE must be used to indentify the unused bit. All output boards must be fully defined ahead of input boards. Outputs or inputs that are unused at the end of a control or indication word need not be defined.

NOTE

Input boards are scanned at a 50 millisecond rate. Control Delivery times are selectable (compiler switch "\$0", refer to section 4.2.2.3).



c. Serial Communications Interfaces

(1) Master Port

The Master serial port (see Figure 4-1) enables a given GENISYS or MICROLOK PLUS unit to communicate with another such Slave unit or units. The MASTER subsection defines the I/O operations for the Master unit of such a system.

For each Slave unit in this system, one ADDRESS...OUTPUT...INPUT group must be defined under the MASTER interface section of the program. The Master unit communicates among the Slave units defined in a round-robin polling scan. The order of the address statements in the program (not the address numbers) represents the order in which Slave units will be polled. The scan time of the Slave units varies, based on the total system configuration. This includes baud rate and number of defined Slave units. The Master unit delivers an output bit to each Slave when the output bit changes value. Note that the connection from a Master port is always made to a correspondingly programmed set of Slave ports. All addresses defined under the MASTER interface section must be in the range of 1-255 (0 is not valid). Each address must also be unique.

(2) Slave Port

The Slave interface on the GENISYS or MICROLOK PLUS unit is designed to communicate with only one such protocol Master unit. In the SLAVE section of the program, only one ADDRESS...OUTPUT...INPUT group may be specified. (Each unit can only be a Slave to one Master unit, but can be a Master to more than one Slave units). Each Slave port only responds to its one, proper address. Therefore, only one ADDRESS group is permitted. When defining the Slave port address in the software, use a number other than zero. Otherwise, set this value to zero and select the address with switch #5 of the Controller board (refer also to section 8.4). As with the other INTERFACE sections, all output bits must be specified before any input bits.

(3) Master/Slave Programming Example (See Figure 4-1)

In this example, the #1 GENISYS or MICROLOK PLUS unit receives two bits (RCV.A.2 and RCV.B.2) from the #2 unit, and two more bits (RCV.A.3 and RCV.B.3) from unit #3. Two bits (SND.A.2 and SND.B.2) are also sent from unit #1 to unit #2, and two bits (SND.A.3 and SND.B.3) are sent from unit #1 to unit #3.

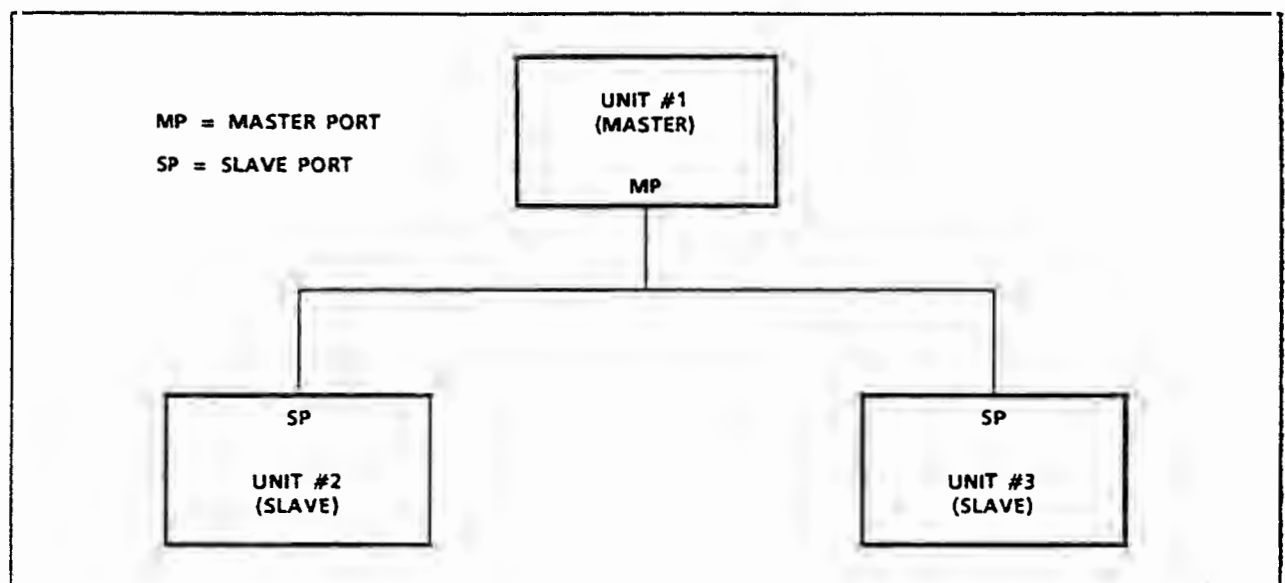


Figure 4-1. Master/Slave Communications Programming Reference Diagram

The program code for unit #1 in Figure 4-1 (MASTER I/O subsection) is as follows:

```
MASTER
ADDRESS:2
  OUTPUT:  SND.A.2, SND.B.2;
  INPUT:   RCV.A.2, RCV.B.2;

ADDRESS:3
  OUTPUT:  SND.A.3, SND.B.3;
  INPUT:   RCV.A.3, RCV.B.3;
```

In the Slave units, the SLAVE subsections must have ADDRESS...OUTPUT... INPUT listings that correspond to the equivalent listing in the Master unit program. Both programs in this example are similar, accepting two input bits and sending two output bits. The input and output bits in the Slave unit programs need not use the same names as the corresponding bits defined in the Master unit program. However, the input/output order must be correct: The first bit on the Master unit's MASTER output list is the first bit on the Slave unit's SLAVE input list, etc.

The program for the Slave section of unit #2 is as follows:

```
SLAVE
  ADDRESS:  2
    OUTPUT:  OUT.2.A, OUT.2.B;
    INPUT:   IN.2.A, IN.2.B;
```

The program for the Slave section of unit #3 is as follows:

```
SLAVE
  ADDRESS:  3
    OUTPUT:  OUT.3.A, OUT.3.B;
    INPUT:   IN.3.A, IN.3.B;
```

In this example, the Master unit (refer to program segment on previous page) outputs "SND.A.2" and "SND.B.2" to Slave unit #2, where they are input into the relays "IN.2.A" and "IN.2.B", respectively. Unit #2 outputs "OUT.2.A" and "OUT.2.B", which are received by the Master unit as "RCV.A.2" and "RCV.B.2". The I/O for unit #3 is similar.

NOTES:

A single GENISYS OR MICROLOK PLUS unit may be programmed to use both its Master and Slave ports simultaneously.

Two or more units programmed to be Slaves of some Master cannot conduct direct serial communication with each other. Any communication between Slave units must pass through their common Master unit.

4.2.4.3 "VAR" Section

The VAR section of the GENISYS program is used to specify the names of internal relays (those that are neither input or output). Relays not defined in the VAR section must be defined as an input or output on a MASTER, SLAVE LOCAL interface. The format of the VAR section is as follows:

```
VAR
  <id list> ;
```

Each identifier in the <id list> is separated by a comma.

4.2.4.4 "TIMER" Section

NOTE

Refer to section 5.2 for a detailed description of system internal timing.

The TIMER section of a program is used to give distinct set (pick) or clear (drop) delays to an internal relay or output bit. A bit not specified in a TIMER statement will, in effect, set or clear instantaneously. Timer delays can be individually specified in the program. The available units include:

1. MIN (minutes)
2. SEC (seconds)
3. MSEC (milliseconds)

Generally speaking, the smaller units have greater timing accuracy:

<u>Unit</u>	<u>Accuracy</u>
MIN	(+/- 3 sec.)
SEC	(+/- 50 msec.)
MSEC	(+/- 5 msec.)

The shortest non-zero delay allowed is 10 milliseconds; and the longest is 25 minutes. The actual ranges that the compiler uses are as follows:

<u>Unit</u>	<u>Range*</u>
MIN	0.1 min. to 25 min.
SEC	0.1 sec. to 25 sec.
MSEC	10 msec. to 2500 msec.

*NOTE

Although the lowest timing ranges for minutes and seconds are given as 0.1 min. and 0.1 sec., respectively, the actual set and clear times specified must be an integer value. For example, 0.1 sec. would be specified as 100 msec., and 0.1 min. as 6 sec.

Values greater than the above may be used. The compiler will automatically convert out-of-range values to the smallest acceptable range. For example, if 3000 MSEC is entered, the compiler will convert this to 3:SEC (with an accuracy of +/-50 milliseconds). However, small values with larger units will not be converted to the smallest units, even if this is possible. For example, if 2:SEC is entered, the compiler will read this as 2 seconds, and not convert this to 2000:MSEC.

The standard format for entering timer values is as follows:

TIMER

```
  { id list } : SET = { integer } : { unit }    CLEAR = { integer } : { unit } ;
  { id list } : SET = { integer } : { unit }    CLEAR = { integer } : { unit } ;
  .
  .
  .
```

Where:

- `<id list>` is a list of previously defined internal or output relay bit names separated by commas.
- `<integer>` is an integer constant in the range of 0 - 9999
- `<unit>` is specified as one of the Reserved Words:
 1. MSEC - for milliseconds
 2. SEC - for seconds
 3. MIN - for minutes

For example:

TIMER

```
T1, T2:  SET = 500:MSEC      CLEAR = 500:MSEC;
1XXRR:   SET = 250:SEC       CLEAR = 0:MSEC;
```

4.2.4.5 Main Program Body

The actual system logic is written in the main program body. Every internal or output relay bit name defined in the INTERFACE and VAR sections should be given a value in the main program body (except "Spare" (keyword) bits). This is done by making the bit the object of an ASSIGN statement (refer to section 4.2.4.6). If an output bit is not the object of an ASSIGN statement, the program may operate properly, but that bit will be ignored. Serial input bits (from Master or Slave) may be the object of an ASSIGN statement. For a complete explanation of how the actual input/logic/output cycle occurs in relation to input assignments, refer to section 4.2.5 (Run Time System Description).

4.2.4.6 ASSIGN Statement

The ASSIGN statement enables the various Boolean operators (OR, NOT etc.) and bit names to be combined in logic equations. For example:

```
ASSIGN IN.A XOR IN.C AND (NOT IN.A OR IN.B)
TO OUT.5;
```

Four logical operators are available to create expressions in a program. These are listed in Table 4-7:

Table 4-7. Logical Operator Symbols

Reserved Word	Shorthand Operator	Reserved Word	Shorthand Operator
NOT	~	OR	+
AND	*	XOR	@

Either form (Reserved Word or shorthand operator character) may be used in assignment expressions. For example, the ASSIGN statement shown on the previous page may be written using shorthand symbols:

ASSIGN IN.A @ IN.C * (~ IN.A + IN.B) TO OUT.5;

The operators AND, OR, NOT and XOR, along with their shorthand symbols, are evaluated according to the truth tables in Table 4-8:

Table 4-8. ASSIGN Operators Truth Tables

Inputs		AND	OR	XOR	NOT	
A	B	A * B	A + B	A @ C	~ A	~ B
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	1
1	1	1	1	0	0	0

The order of evaluation is determined by the following precedence rules:

HIGHEST: NOT AND
LOWEST: OR XOR

Operations with the highest precedence are performed first. Operations at the same precedence level are evaluated left to right. Parentheses may be added to alter this default order. Parentheses take precedence over the defined operational order. The order of evaluation of the above assignment statement is shown in Figure 4-2:

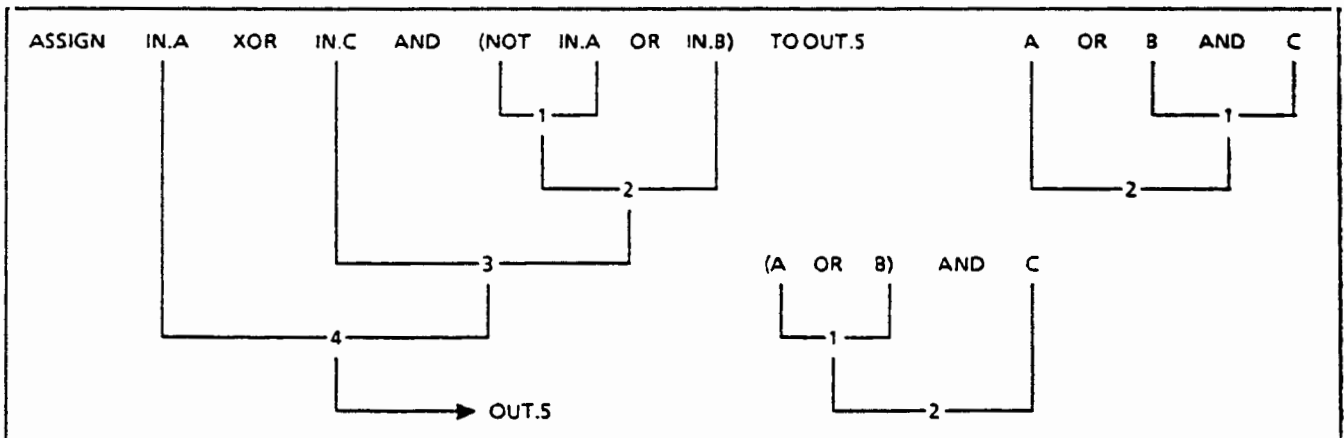


Figure 4-2. ASSIGN Operators, Order of Precedence Samples

If more than one relay is to be assigned the same value, the logic expression need not be repeated in a second ASSIGN statement. Additional relay names may be assigned using the same statement by listing them after the "TO" Reserved Word, separated by commas. For example:

ASSIGN A AND B OR NOT C TO D,E,F;

4.2.5 Run Time System Description

4.2.5.1 Input/Output Description

The GENISYS and MICROLOK PLUS (non-vital section) systems use an internal RAM table containing the current state of every relay defined in the system. This table includes the LOCAL input and output bits, as well as the internal bits defined in the VAR section.

The values of LOCAL input bits in the table are automatically updated in a 50 millisecond scan. Therefore, LOCAL inputs may not receive a value based on an ASSIGN statement, otherwise a logical conflict could exist. For example, there could be an input with an actual state of "1", and a logic ASSIGN statement which computes its value as "0".

The LOCAL output bits are delivered in groups of 16 whenever any one of the bits in that group change state. If Constant Delivery relay-output boards (N451441-7101) are used, the outputs directly follow the states of the internal table.

Input bits on either the Master or Slave serial line have their internal table value updated differently. When a GENISYS or MICROLOK PLUS unit receives a bit on a serial line, its received value is put in a table. It will keep that value until the bit is received again. Because the input value of serial bits is only updated when received, each may also be given a value with an ASSIGN statement. However, no more than one ASSIGN statement may be specified for each relay.

(The slot off of a signal request application illustrates a serial input bit being given a value from an ASSIGN statement. The controlling Master unit, which may be a full office mini-computer, may send a green request using a serial input bit. This request may be combined with other information to produce the actual green output. The green signal request is usually to be cancelled when the appropriate track circuit becomes occupied. Note that logic may be written to clear the signal request input from the serial line.)

4.2.5.2 Logic Processing

Note

Refer to section 5.4 for a detailed description of logic processing and queuing.

An ASSIGN statement is equivalent to tracing the path or paths of current flow from a battery to the assigned relay coils. Therefore, the logic for all of the wiring associated with that relay coil must be contained in one ASSIGN statement. The order of the ASSIGN statements in the source program will usually have no effect on the output.

An ASSIGN statement is re-computed each time one of the relay coils included in the statement changes state. This involves the following chain of events: The system first determines if the coil has a timing delay. If so, the specified time is run until the relay is to be set or cleared. Assignments for non-timer coils are carried out immediately, and all ASSIGN statements in

which this coil is referenced are re-computed. This process continues until the system reaches a stable state. During the time ASSIGN statements are being computed, no output operations (local or serial) are performed; inputs are still scanned at 50 millisecond rate. Any output started before the logic sequence will continue to be processed. However, outputs cannot be updated.

In accordance with actual relay-logic practices, logic processing always uses a break-before-make format: When a relay changes values, the break is done before the make. However, there are circumstances where a make may occur before a break in a relay chain, depending upon the setting of the queueing compiler switch $\$QV$ (v = + or -). The relay execution order in Figure 4-3 shows the operation of this switch option:

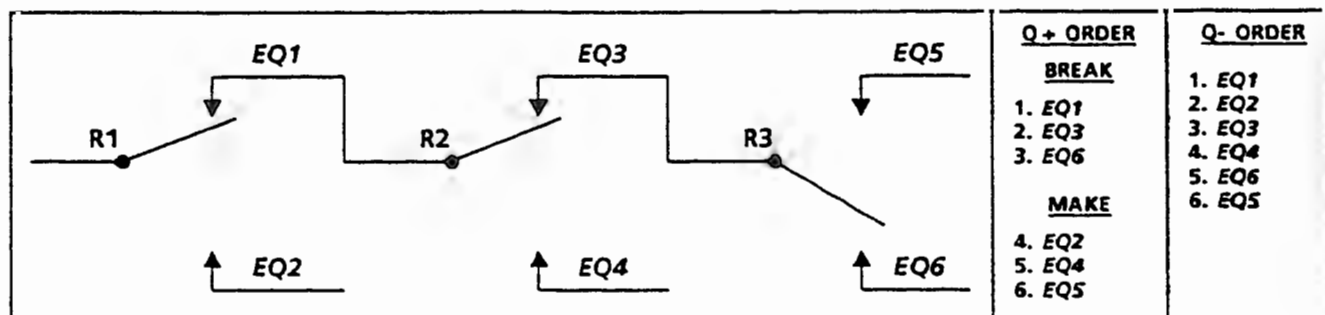


Figure 4-3. Queueing Option Reference Diagram

When the switch is set to Q- (default), equations are processed relay by relay, with breaks executed before makes in the order in which they occur. In Figure 4-3, if relay R1 changes state, EQ1 is executed (break), followed by EQ2 (make). Next, the change in EQ1 (relay R2) causes EQ3 to be executed (break), thus the make of EQ2 occurs before the break of EQ3.

If the queueing switch is set to Q+, there are two queues for logic equations, including one for the breaks and one for the makes. Breaks are always done first, regardless of when they happen. In Figure 4-3, note that the three break equations (EQ1, EQ3, EQ6) are executed before any of the makes. Even if the makes were queued first and then break occurred, the break is done first.

NOTE

The following paragraph applies to GENISYS Development System versions 3.0 and higher:

In versions of the compiler before 3.0, the Break-Before-Make rule was not always followed when processing some types of logic equations. Under certain limited conditions, logic equations could be executed in an order that would not strictly follow the Break-Before-Make ordering. Version 3.00 is designed to assure that all equations will be properly processed and executed in the Break-Before-Make order.

If an older application logic program is recompiled with version 3.00, the EPROMs generated may not be identical to the older EPROMs. These differences would only involve the order in which the logic equations are processed. No changes are required on any installation that is currently in service and has undergone complete testing during a cut-over.

NOTE

The following paragraph applies to GENISYS Development System Versions 1.03 and higher, and Executive PROM IC29 Revisions 3 and higher. In earlier Versions and Revisions, a maximum of 500 equations may be queued at any one time. If the 2-queue option (Q+) is used, a maximum of 250 equations may be present on each queue (break and make).

A maximum of 1000 equations may be queued at any one time. If the 2-queue option (Q+) is used, a maximum of 500 equations may be present on each queue (break and make). Any attempt to queue more equations, using either option, will be ignored by the Run Time executive. The Q- option is slightly faster and should be used when possible. However, it may cause problems with some stick circuits (consult US&S Engineering).

4.2.5.3 Serial Communications - Pre-Defined Relays

When the serial I/O lines (Master or Slave) are used in GENISYS or MICROLOK PLUS, it may be necessary to condition logic processing for successful or unsuccessful completion of communications (i.e. code line up or down). For example:

1. A Master unit may be programmed to turn on an output (i.e. an indicator lamp) when it loses communications with a Slave unit.
2. A Slave unit may be programmed to perform field "auto-clear" functions if communications are lost with its Master unit.

Both of these situations can be accommodated with the GENISYS or MICROLOK PLUS (non-vital) system. The compiler contains pre-defined relay names that are SET and CLEARED automatically by the Run Time system.

On the Master Port, one internal relay is created for each Slave unit address specified in the MASTER part of the INTERFACE section. Each name is automatically defined as follows:

SLAVE.ON.n

The letter "n" corresponds to the address of the particular Slave unit served by the relay. The value of "n" may range from 1 to 255, the same range as Slave unit addresses.

These relays may be used as part of an expression in an ASSIGN statement to condition logic, or may be listed in an output statement. Since they have a predetermined value, they may not appear in an input statement, or be the object of an ASSIGN statement.

Initially, these relays are clear (dropped). When the first valid message is received from a Slave unit, the associated relay in the Master is SET (picked). The relay will remain SET until the Master detects two consecutive errors in transmission. At this time, the relay is cleared (dropped) and will remain in this state until communications are resumed with that Slave.

At each Slave unit, one internal communication relay is automatically defined as:

MASTER.ON

This relay is automatically SET and CLEARED by the Run Time system to indicate the status of communications from the Master unit. It may be used as part of an expression in an ASSIGN statement to condition logic, or may be listed in an output statement. Since it has a pre-defined value, it may not appear in an input statement, or be the object of an ASSIGN statement.

The MASTER.ON internal relay is SET and CLEARED differently from the SLAVE.ON.n relays. MASTER.ON is initially CLEAR (dropped). When the Slave unit receives the first valid message addressed to its station number, the bit is set. The Slave unit then continues to look for the text header, followed by a trailer, followed by text header, etc. If more than five seconds elapses after receiving (1) a header with no trailer, or (2) a trailer without a new message that starts with a header, the bit is cleared.

4.2.5.4 Valid Bit Option - Introduction

NOTE

Refer to section 5.3 for a detailed description of the Valid Bit Option.

At power-on reset, the GENISYS or MICROLOK PLUS (non-vital) unit sets all internal and output relays to zero (clear). No actual output can occur at this time. After the initial reset, optical-input boards (if any) are scanned to determine the value of bits defined as LOCAL INPUT bits in the program. If any input bits are defined in the program, but refer to boards not actually in the cardfile, those bits will be given "Invalid" or unknown values. Similarly, any bit defined as an input on either of the two serial lines becomes "invalid" until the bit is received from the unit at the other end of the line.

During processing of an ASSIGN statement, any "invalid" input bit referenced during the computation will stop the processing procedure. The relay defined as the object of this ASSIGN statement cannot be changed. Only "invalid" input bits will stop the processing of a logic statement.

Output and internal bits may also be "invalid", but will not interrupt the completion of an ASSIGN statement. At reset, these types of bits are initialized as zeros. They are treated as valid zeros during processing of ASSIGN statements. Once all output bits in a LOCAL output word or serial protocol byte become valid, the LOCAL word is delivered to the actual outputs, or it becomes available to the serial line. However, the bits cannot be delivered until they have received a "valid" value as the result of successfully resolving an ASSIGN statement.

NOTE

The following statement applies to GENISYS Development System Versions 1.02 and higher. In Versions 1.00 and 1.01, the indicated Boolean identities would not be evaluated.

The following Boolean identities will be evaluated even if x has an "invalid" value:

$$1 \text{ OR } x = 1$$

$$0 \text{ AND } x = 0$$

In these identities, the value of x is irrelevant. 1 Ored with anything is a 1, and 0 Anded with anything is a 0.

4.2.6 Relay Models and Programming Techniques

Figure 4-4 provides sample relay circuits for GENISYS and MICROLOK PLUS (non-vital) programming models. The flasher relay set-up could not exist in actual relay logic, but is possible in GENISYS or MICROLOK PLUS by establishing a distinct pick-up/drop-away interval for the relay. The pertinent parts of the program include:

T1: SET=1:SEC CLEAR=1:SEC;

ASSIGN NOT T1

TO T1;

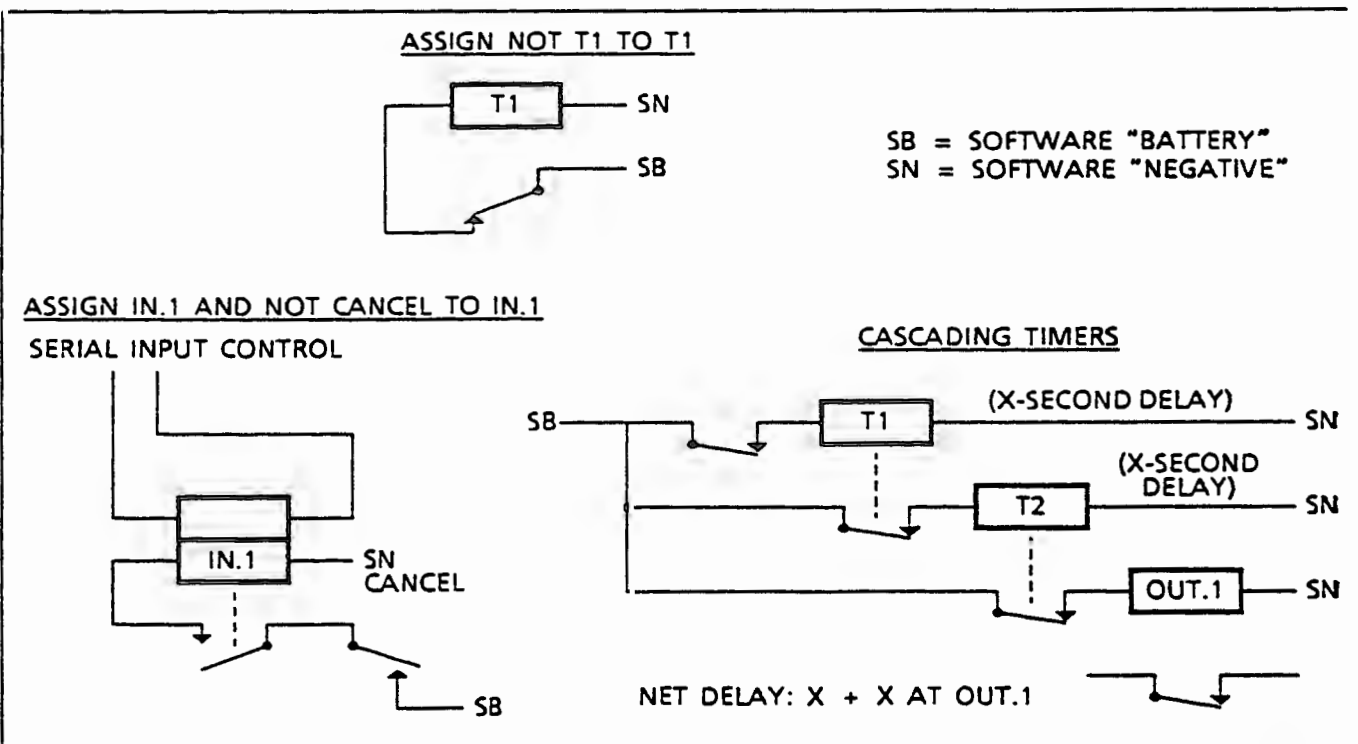


Figure 4-4. Conceptual Relay Models for GENISYS and MICROLOK PLUS Programming

If the timer value is not specified, the "contact" will operate at a speed which cannot be detected by the run time system. This would create an indeterminate function at this point, inhibiting execution of the program.

The double coil relay examples in Figure 4-4 are conceptual models which pertain to signal control slotting. Models can not apply to circuits with control contacts to both coils of a double-coil relay. The pertinent program statement for the model containing the CANCEL contact is:

```
ASSIGN NOT CANCEL AND IN.1          TO IN.1;
```

The capability to assign to serial inputs may be applied to the design of auto clearing controls. In the case of auto clearing of requests, if the input is a request for a clear signal and CANCEL is a track relay, the signal request will be cleared when the track is occupied.

The following statement shows how requests from the office can be cleared if the communications link is lost:

```
ASSIGN MASTER.ON AND IN.1          TO IN.1;
```

(These techniques are presented as conveniences for designing auto clearing logic. They may not be suitable in all applications. Consult US&S Engineering for further guidance in the development of this type of logic.)

Cascading timer relays are used to build up a comparatively long delay on the output delay relay. Only one timing specification (SET and CLEAR) needs to be written for all timer relays (equal addends for the desired delay):

```
T1, T2:  SET=5:SEC          CLEAR=5:SEC
```

4.3 GENISYS DEVELOPMENT SYSTEM (G.D.S) - GENERAL

The GENISYS Development System (G.D.S) enables the user to compose, debug and load an application program into the GENISYS or MICROLOK PLUS (non-vital section) system hardware. Refer to Appendix A for development system components. (Note: A text editor is not available with development system.)

Figure 4-5 shows the general steps in the use of the G.D.S. The source program is written and entered into the compiler using a text editor. The compiler checks the program for proper terminology and format and lists any errors in a listing file. Using the information in this file, the user returns to the text editor and corrects these errors. The compiler cannot detect mistakes in the user's application of logic statements. These are located using the Simulator. Again, the user returns to the text editor to correct errors. When the simulator indicates satisfactory operation of the program, it may be loaded into the Controller PCB EPROM. The compiler is used to convert the source program into PROM tables. The EPROM programmer unit performs final checks of these tables and the EPROM itself, before actual loading of the data into the chip.

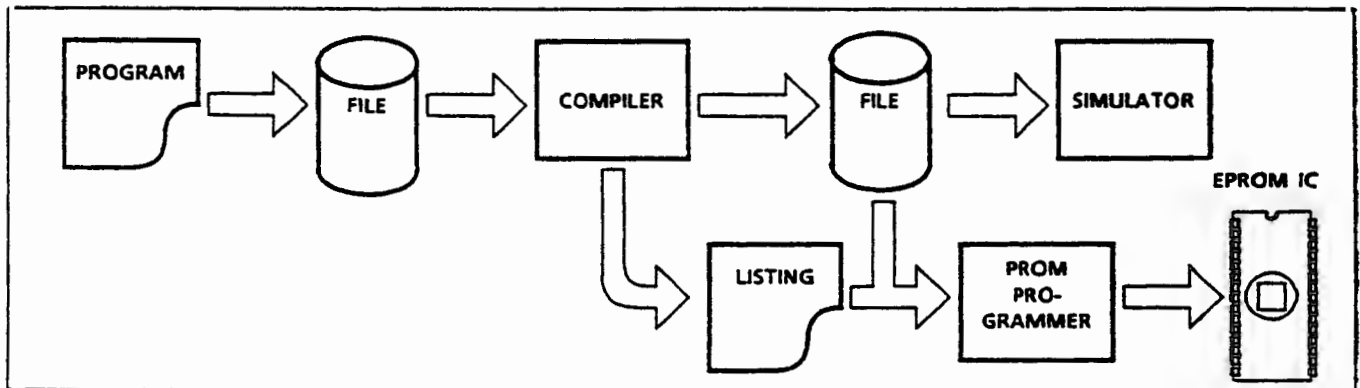


Figure 4-5. Development System Block Diagram

4.4 G.D.S. - AVAILABLE FILES

The GENISYS Development System software uses seven file extensions. These extensions enable the user to employ the various parts of the G.D.S:

<u>Extension</u>	<u>File Contents</u>
.GEN	Source program for the application logic
.GLS	Listing file with errors produced by the compiler.
.GID	Identifiers file produced by compiler and used by simulator.
.GCD	EPROM code file produced by the compiler/assembler and used by the simulator and EPROM programmer.
.GEQ	Equations generated by the simulator
.GSI	Simulator initialization file.
.GIM	Temporary file used during compilation.

4.5 G.D.S. - COMPILER

NOTE

Different revisions of the GENISYS Development System compiler may yield different check sums and code sizes. However, the program should behave in the same manner.

The compiler checks and converts the application program into a code that can be processed by the Executive software (contained in a separate Controller PCB EPROM). The compiler performs two functions, including code generation and assembling.

The code generation section checks the source program for any errors, using a two-phase process: Syntax Analysis and Semantic Analysis. Syntax analysis looks for improper "grammar" in the program. During this analysis, two types of errors may be detected, including token errors (more than 12 characters, illegal character, etc.) and statement syntax errors (no BEGIN Reserved Word, missing semicolon, etc.).

Semantic analysis checks for meaningful statements. For example:

```
      ASSIGN A AND B                                TO C;
```

If C is defined as an input bit, or B has not been defined, a semantic error will be detected. Refer to sections 8.1 and 8.2 for a complete listing of compiler error messages.

As the source program is processed, it is converted to a special-purpose code that, in turn, is processed by the Assembler section of the compiler. The assembler converts the output of the code generator to a format that can be processed by the EPROM Programmer.

The compiler is accessed by using the batch file: GENISYS "name". Typing in this term invokes the batch file. This file can perform several functions, depending on how it is invoked:

GENISYS name	-If <name> .GEN appears as a file in the current directory, it is compiled. Otherwise, a "file does not exist" message is displayed.
GENISYS HELP	-This displays the G.D.S. help file. This file is shown at the top of the next page.

The batch file performs several operations:

1. Determines if the requested file exists; displays error message and stops if this file does not exist.
2. Deletes the previous EPROM code (.GCD) file
3. Calls the code generator
4. Calls the assembler if no errors were detected.

The batch file is recommended for compiling a GENISYS or MICROLOK PLUS non-vital program.

Version 3.00

- NOTE**

4.6 G.D.S. - SIMULATOR

The GENISYS Simulator allows testing of a completed GENISYS or MICROLOK PLUS non-vital program, prior to actual loading into the system hardware. Commands are provided in the simulator to mimic operating aspects of the designed system. This includes setting and clearing internal and external relays, executing logic equations, and advancing system time. Logic statements and the system clock can be stepped individually, or simultaneously at any desired increment. Commands are also available to display: (a) inputs and outputs according to their actual arrangement in the system cardfile, (b) names, bit numbers and status of individual relays, (c) logic statements as they are executed.

6300A, D. 4-28

The simulator supports the following two switches:

<u>Switch</u>	<u>Name</u>	<u>Comments</u>
\$Q	Queuing Option	If \$Q+, use two queues (break before make), otherwise use only one queue. (Default is Q-, one queue.)
\$V	Validation Check	If \$V+, INVALID becomes a valid state, and is checked during execution of logic equations. If an equation contains an invalid bit, it will be executed per the rules in section 4.2.5.4. If \$V-, all bits are cleared. (Default is V+, validation on.)

4.6.2 Access to Simulator

4.6.2.1 General

NOTE

The following paragraph applies to GENISYS Development System Versions 1.01 and higher. Refer to section 4.2.2.3, page 4-6, for the debug switch used in Version 1.00.

The simulator uses the EPROM file produced by the compiler. If a program is to be debugged using the simulator, it must be compiled with the debug switch on ($\%\$D+\backslash$). The program does not have to be recompiled with $\%\$D-\backslash$ to permit programming of PROMs. The default is $\%\$D-\backslash$.

Command terms in the following text are shown in all capital letters, to help distinguish them from other words in the text. In the actual use of the simulator, these words may be typed in lower case letters as well as upper case. <CR> indicates the carriage return key (or ENTER).

4.6.2.2 Procedure

NOTE

In GENISYS Development System Versions 1.02 and higher, the user can enter the simulator as "GENSIM" or "GENSIM <file> ". The latter option allows the file name to be obtained from the command line, without being prompted.

1. The simulator is run by entering GENSIM and a carriage return (<CR>), as shown in the Help File. The simulator cover screen will then be displayed, showing the version of the simulator.
2. The prompt on the cover screen asks for the name of the source program. When the name is entered, a tabulation will appear immediately below. This table lists the basic totals of bits and boards in the source program.

3. Entry of the program name will also produce several permanent "status" lines near the bottom of the screen. These lines are explained in section 4.6.3. To obtain a summary of all simulator commands, enter HELP (CR) at the command prompt in the second status line. The Help Screen is explained in section 4.6.4.3.

NOTE

The examples shown use programs compiled with the Validation Option on (%\$V+ \). This initially flags all bits as invalid.

4.6.3 Standard Formats

A typical set of status lines for the beginning of a simulator exercise is shown below:

Trigger List: 10	System Time: 00:00:00:000	Timer List: 0
Command-	Program: NBR3	Screen: Init

The "Trigger List" refers to the total number of logic equations that are queued to be executed. Whenever a bit changes, all logic equations that use that bit will be placed on the Trigger List.

The "System Time" records the total time elapsed during the execution of simulation. Left to right, increments are in hours: minutes: seconds: milliseconds. All timed operations are advanced in increments of tens of milliseconds.

The "Timer List" provides the total number of timer relays in the source program that are on a timer queue.

Command names and lists of items such as relays are entered after the "Command" prompt, and executed with a carriage return (CR). When entering a series of non-consecutive items after the command, leave a space between each item number (i.e. 1 3 7 12 19). When entering a series of consecutive items, a dash may be used for the intermediate items (i.e. 1-5 7-9 11-30). Use the same procedure when listing items by name (i.e. OUT.1-OUT.3 OUT.5). If more items are requested than are available in the listing, an "invalid range" error listing will appear. Do not attempt to enter more items than space allows between "Command" and "Program". Instead, enter a (CR) after the first group and repeat the command to enter a following group.

"Program" indicates the name of the source program undergoing the simulation.

"Screen" indicates the type of information presently on the screen. For example, the initial GENSIM cover screen is "Init".

Scrolled information is run in the space immediately below the status lines. For example, selected commands show source program logic equations and output results in this area, as the program is executed. This section shows a maximum of four lines at one time. To scroll information from the top of the screen, enter NO (CR). This is the command for "No Display". The status lines will remain on the screen.

4.6.4 Simulator Operation

4.6.4.1 General

When the simulator is executed and the EPROM tables are entered, the simulator performs the necessary housekeeping functions:

- (a) Initialize all bits (either CLEAR OR INVALID)
- (b) Reset system time
- (c) Initialize all queues and lists
- (d) Queue all equations on the Trigger List

4.6.4.2 Sample Program

Simulator commands are described in subsequent sections, using a sample program shown on page 4-32. Each of the available commands (see Help screen, section 4.6.4.3) is exercised with this program in a typical order of execution, and not in order shown on the Help screen. This is not a required order; with practice, the user will find it desirable to call up a variety of commands at any point in the simulation.

NOTES

Note the MICROLOK PLUS LOCAL I/O assignments at the bottom of the sample program. This shows the 2-board limit placed on the MICROLOK PLUS non-vital section. (Compare this with the GENISYS sample program, which covers four total I/O boards.) Otherwise, the complete MICROLOK PLUS sample program is identical to the GENISYS program, except for the left-hand line numbering.

Certain aspects of the Simulator's operation, such as scrolling lines, cannot be depicted in this text. To best understand the operation of the Simulator, the user should enter the sample program in his system and run the various commands as they are presented.

The card slot locations of the inputs and outputs are shown in the symbol tables on page 4-33 (GENISYS) and 4-34 (MICROLOK PLUS).

NOTE

As indicated in the MICROLOK PLUS diagram, board slot numbers are based on the GENISYS cardfile slot numbering. Make certain to note the difference in the MICROLOK PLUS non-vital section slot numbering.

GENISYS* PROGRAM

GENISYS Source Listing

[Version 3.00]

5-JUN-1991

Page: 1

Copyright 1984, Revised 1991,

Union Switch & Signal Inc.

4
TOTAL
I/O
BOARDS

```

1      %
2      !      THIS IS AN EXAMPLE
3      !      OF A BASIC NON-VITAL
4      !      PROGRAM.
5      !      \
6
7      %$D +      DEBUG ON\
8      %$Q +      TWO QUEUE OPTION\
9
10     PROGRAM NBR3;
11     INTERFACE
12         LOCAL
13             OUTPUT WORD:  OUT.1, OUT.2, OUT.3,
14                             SPARE, SPARE, SPARE, SPARE, SPARE,
15                             OUT.4, OUT.5, OUT.6;
16             OUTPUT WORD:  OUT.7, OUT.8;
17             INPUT WORD:   IN.A, IN.B;
18             INPUT WORD:   IN.C, IN.D;
19
20     VAR STICK, T1;
21
22     TIMER
23         T1:      SET = 1:SEC      CLEAR = 1:SEC;
24         OUT.7 :  SET = 1:SEC      CLEAR = 1:SEC;
25         OUT.8 :  SET = 0:SEC      CLEAR = 500:MSEC;
26
27     BEGIN
28         ASSIGN IN.A AND IN.B      TO OUT.1;
29         ASSIGN IN.A XOR IN.B      TO OUT.2;
30         ASSIGN IN.A OR IN.B       TO OUT.3;
31         ASSIGN NOT IN.A           TO OUT.4;
32         ASSIGN IN.C OR (STICK AND NOT IN.D) TO STICK;
33         ASSIGN STICK              TO OUT.6;
34         ASSIGN NOT T1             TO T1;
35         ASSIGN T1                 TO OUT.7;
36         ASSIGN NOT T1             TO OUT.8;
37     END

```

Errors Detected: 0

LOCAL I/O SECTION OF MICROLOK PLUS™ NON-VITAL PROGRAM

2
TOTAL
I/O
BOARDS

```

12     LOCAL
13     OUTPUT WORD:  OUT.1, OUT.2, OUT.3,
14                     SPARE, SPARE, SPARE, SPARE, SPARE,
15                     OUT.4, OUT.5, OUT.6, OUT.7, OUT.8;
16     INPUT WORD:   IN.A, IN.B, IN.C, IN.D;

```


NOTES

GENISYS[®] SYMBOL TABLE

The sample bit symbol table below applies to GENISYS[®] Development System Versions 1.01 and higher. In Version 1.00, the "BIT TYPE" column is placed in a separate column at the right and the "TIMER" column is called "TIME DELAY".

The Version 1.00 table as a whole covers 132 spaces; in Versions 1.01 and higher, the table covers 80 spaces.

In the "SWITCH SETTINGS" listing at the bottom, Version 1.00 does not include "Polling Time Out" and the "Security" switch is "ON" or "OFF", rather than "ON" or "AUTOMATIC".

4
TOTAL
I/O
BOARDS

Bit Symbol Table for "NBR3"

5-JUNE-1991

PAGE SYM- 1

IDENTIFIER	BIT NUMBER & TYPE	TIMER	
		SET	CLEAR
LOCAL I/O BOARDS			
OUTPUT - SLOT 3			
OUT.1	1 OUTPUT		
OUT.2	2 OUTPUT		
OUT.3	3 OUTPUT		
SPARE	4		
SPARE	5		
SPARE	6		
SPARE	7		
SPARE	8		
OUT.4	9 OUTPUT		
OUT.5	10 UNASSIGNED		
OUT.6	11 OUTPUT		
OUTPUT - SLOT 4			
OUT.7	12 OUTPUT	1.0: SEC	1.0: SEC
OUT.8	13 OUTPUT	0.0: MSEC	500.0: MSEC
INPUT - SLOT 5			
IN.A	14 INPUT		
IN.B	15 INPUT		
INPUT - SLOT 6			
IN.C	16 INPUT		
IN.D	17 INPUT		
LOCAL VARS			
STICK	18 INTERNAL		
T1	19 INTERNAL	1.0: SEC	1.0: SEC

SWITCH SETTINGS

Slave Baud Rate : Hardware switch
Master Baud Rate : 1200 BPS
Polling Time Out : 1000 MSEC
Control Delivery Time : Hardware switch
Security : Automatic
Debug : ON
Symbol Table Listing : ON
Validity Check : ON
Two Queue Option : ON

MICROLOK PLUS™ SYMBOL TABLE

Bit Symbol Table for "NBR3"

5-JUNE-1991

PAGE SYM- 1

2
TOTAL
I/O
BOARDS

IDENTIFIER	BIT NUMBER & TYPE		SET	TIMER	CLEAR
LOCAL I/O BOARDS					
OUTPUT - SLOT 3					
OUT.1	1	OUTPUT			
OUT.2	2	OUTPUT			
OUT.3	3	OUTPUT			
SPARE	4				
SPARE	5				
SPARE	6				
SPARE	7				
SPARE	8				
OUT.4	9	OUTPUT			
OUT.5	10	UNASSIGNED			
OUT.6	11	OUTPUT			
OUT.7	12	OUTPUT	1.0: SEC	1.0: SEC	
OUT.8	13	OUTPUT	0.0: MSEC	500.0: MSEC	
INPUT - SLOT 5					
IN.A	14	INPUT			
IN.B	15	INPUT			
IN.C	16	INPUT			
IN.D	17	INPUT			
LOCAL VARS					
STICK	18	INTERNAL			
T1	19	INTERNAL	1.0: SEC	1.0: SEC	
SWITCH SETTINGS					
Slave Baud Rate : Hardware switch					
Master Baud Rate : 1200 BPS					
Polling Time Out : 1000 MSEC					
Control Delivery Time : Hardware switch					
Security : Automatic					
Debug : ON					
Symbol Table Listing : ON					
Validity Check : ON					
Two Queue Option : ON					

NOTE

"OUTPUT - SLOT 3" is slot "P" in the MICROLOK PLUS™ cardfile.
"INPUT - SLOT 4" is slot "Q" in the MICROLOK PLUS™ cardfile.

4.6.4.3 Help Screen

HELP <CR> produces the standard Simulator Help screen shown below. This screen shows all display and operating commands, and their purposes. Note that the command names include capital and lower case letters. The capital letters are the minimum characters required for a valid command. Using the QUIT command as an example, Q <CR> would cause an error message, but QU <CR> would be a valid command. (QUI and QUIT are also valid.) Capital letters are only used on the help screen to show the shortest substring that can be specified for that command. In practice, these may also be entered in lower case letters (for example, qu, qui, quit).

NOTE

The No Display command is discussed where it might be used with some of the other commands.

*****HELP SCREEN*****			
Display:	DISplay	{IO}	display all values of I/O boards
		{TRiggers}	display all equations on trigger list
		{RElays} [(list)]	display all relays on display list
		{Timers}	display all relays on timer queue
	VALue (list)		display value of relay(s)
	REMOve (list)		remove relays(s) from list
	NOdisplay		full screen display
	COLor		use color characteristics for display
	MOno		use monochrome characteristics for display
Simulation:	RUn [x]		run system for x milliseconds
	EXecute [x]		execute x number of logic equations
	INCrement [x]		increment system clock x milliseconds
	TRace [x]		display and execute x logic equations
Bit Operations:	SEt (list)		set relay(s) (list)
	CLear (list)		clear relay(s) (list)
	INPut (list)		input values for board(s) (list)
Control:	PRint {file}		print all logic equations
	REAd {file}		read commands from file
	QUIT		end simulation
	RESet		reset system (all bits invalid)
	HElp		this screen
Trigger List: 0 , 0		System Time: 00:00:00:000	
Command-Help		Program: NBR3	
		Timer List: 0 , 0	
		Screen: Help	

The first four display commands require a combination of DI, a space, and the minimum letters of the command. For example DI TR <CR> would display all equations currently on the trigger list.

The notations after the Help screen commands are defined as follows:

- [] Optional argument. If this is not specified, the simulator will resort to the associated default. For example, if the Display Relays command is entered without a list of requested relays, all relays on the relay display list will be shown.
- (list) List of bit names/numbers. The list is used to specify a bit or series of bits (refer also to section 2.6.3).
- { } Required argument. If not specified, it will be prompted.
- file File name. Default extensions will be added.

4.6.4.4 Display IO Command

The Display I/O (DI IO <CR>) command shows the physical arrangement of the source program inputs and outputs on the cardfile I/O boards. In the example at the top of the following page, Out #1 etc. represents the relay-output PCB(s) and their word numbers in the cardfile. Like the actual cardfile installation, relay outputs are located to the left of the optical-inputs. Note the difference between the GENISYS and MICROLOK PLUS displays. The 1-16 column on the left represents the output and input bit numbers, respectively, on the relay and opto boards. (Make certain not to confuse these with the 0-15 output and input bit numbering on the PCBs themselves. Bit 1 on this screen is equivalent to bit 0 on the PCB.). The line dividing bits 1-8 and 9-16 represents the two 8-bit bytes transferred consecutively off the PCB.

At the start of a simulation, question marks (?) will appear at the specific bits of each board if the validate value compiler switch is %\$V+\. (If the switch is set to %\$V-\, all bits equal 0.) At this time, all active inputs are invalid (neither 0 or 1), and are represented by this symbol. Any SPARE relays are indicated as clear ("clr"), but will have no effect on the program. As the simulation is run, the invalid relays will change to "clr" (clear) or "set" (set).

4.6.4.5 Display Triggers Command

The Display Triggers (DI TR <CR>) command gives a listing of all equations on the trigger list, by specifying the line number generated by the compiler. The screen from the sample NBR3 program is shown at the middle of the following page. The tabulation at the bottom of the next page may be used to determine which list an equation will be added to when a bit changes. It is also possible for an equation to be queued on both lists. In the NBR3 example, line 26 appears on both lists:

```
26          ASSIGN IN.C OR (STICK AND NOT IN.D)          TO STICK;
```

In this equation (both IN.C and IN.D start out with a value of zero), "NOT IN.D" is queued on the make list and IN.C is queued on the break list. This equation will be executed twice: once from the Break list and then from the Make list. (If switch %\$Q-\, only one trigger list exists.)

Initial NBR3
Display I/O
Screen for
GENSYS®
Program

	OUT #1	OUT #2	IN #1	IN #2		OUT #1	IN #1
1	?	?	?	?	1	?	?
2	?	?	?	?	2	?	?
3	?				3	?	?
4	clr				4	clr	?
5	clr				5	clr	
6	clr				6	clr	
7	clr				7	clr	
8	clr				8	clr	
9	?				9	?	
10	?				10	?	
11	?				11	?	
					12	?	
					13	?	

Equivalent
listing for
MICROLOK
PLUS™

Trigger List: 10
Command-> DI TR

System Time: 00:00:00:000
Program: NBR3

Timer List: 0
Screen: I/O Boards

Initial NBR3
Trigger List
Screen

MAKE	List	BREAK	List
Line	22	Line	25
Line	23	Line	26
Line	24	Line	28
Line	26	Line	30
Line	27		
Line	29		

Trigger List: 10
Command-> DI TR

System Time: 00:00:00:000
Program: NBR3

Timer List: 0
Screen: Triggers

Trigger List Development Table

Bits Value	Changing To	Equation	
		Uses	Queued On
Bit 0	1	bit	Make List
Bit 0	1	~ bit	Break List
Bit 1	0	bit	Break List
Bit 1	0	~ bit	Make List

4.6.4.6 Display Relays Command

The Display Relays (DI RE ____ (CR)) command has two forms: selected relays and all relays.. For example, DIS RE 1-3 9 14-17 (CR) adds relays 1 through 3, 9, and relays 14 through 17 to the list. DI RE (CR) shows all relays currently on the list. The present state of the relay (set, clr, ?) is shown in the value column. If the relay has a pick (set) or drop (clear) delay, its state will be displayed in the status column. All times are specified in milliseconds. As with the Display IO command, all active relays begin with an invalid (?) value and all SPAREs are shown as clear ("clr").

Up to 34 relays may be displayed at any given time. The relay display list can hold up to 34 relays. If more relay additions are attempted to a full display list, an error message will be generated. When spare relays are present on an initial listing, they can be removed to allow display of more active relays on a given screen. This is done with the Remove command; refer to section 2.6.4.7. In the following example, DI RE 1-19 (CR) was entered:

-Display All Relays-

Bit Number - Name	Value	Status	Bit Number - Name	Value	Status
1 OUT.1	?		18 STICK	?	
2 OUT.2	?		19 T1	?	
3 OUT.3	?				
4 SPARE	clr				
5 SPARE	clr				
6 SPARE	clr				
7 SPARE	clr				
8 SPARE	clr				
9 OUT.4	?				
10 OUT.5	?				
11 OUT.6	?				
12 OUT.7	?				
13 OUT.8	?				
14 IN.A	?				
15 IN.B	?				
16 IN.C	?				
17 IN.D	?				

Trigger List: 10
Command-> DI-RE 1-19

System Time: 00:00:00:000
Program: NBR3

Timer List: 0
Screen: Relays

Relays may also be displayed by name. For example, DI RE IN.A (CR) would display the same relay as DI RE 14 (CR) (bit #14 and IN.A refer to the same relay).

4.6.4.7 Remove Command

The Remove (REM ____ <CR>) command allows removal of relays from the relay display list. Relays can be removed by bit number or name. In the example below, REM 4-8 <CR> was entered to remove spare relays from the Display Relays table in the previous section. If relay OUT.4 had been accidentally removed from this listing, it could be restored with the command DI RE OUT.4 <CR>. However, OUT.4 would reappear at the end of the listing, rather than in its numerical position. If the simulator is exited or reset, the SPARE relays will appear again on this listing, and must be removed again.

-Spare Relays Removed-

Bit Number - Name	Value	Status	Bit Number - Name	Value	Status
1 OUT.1	?				
2 OUT.2	?				
3 OUT.3	?				
9 OUT.4	?				
10 OUT.5	?				
11 OUT.6	?				
12 OUT.7	?				
13 OUT.8	?				
14 IN.A	?				
15 IN.B	?				
16 IN.C	?				
17 IN.D	?				
18 STICK	?				
19 T1	?				

Trigger List: 10
Command-> REM 4-8

System Time: 00:00:00:000
Program: NBR3

Timer List: 0
Screen: Relays

4.6.4.8 Input Command

The Input (INP ____ <CR>) command is used to set or clear the input bits in the source program. Either a single number or range of numbers may be entered with this command, representing the number(s) of the input boards. In the example at the top of the following page, the Display IO screen is called up and INP 1 <CR> is entered. Then a 0 <CR> is entered.

In the scroll area, the names and numbers of all relays on the first optical-input board (to the right of the relay boards) are listed, along with the instruction for entering a logic 0 or 1. The entered 0 changes the state of the first input bit on the #1 opto board from invalid (?) to clear (clr). A logic "1" sets that relay. When all input bits have been defined for opto board #1, the command stops. This command also accepts a range: INP 1-2 would cause the simulator to input values for board #1, followed by board #2.

Once the input bits have been defined, they cannot be returned to the invalid state without exiting or resetting the program.

-Input 1, (Board #1) cleared-

Input 1 cleared
in GENISYS®
program.

	Out #1	Out #2	In #1	In #2
1	?	?	dr	?
2	?	?	?	?
3	?			
4	dr			
5	dr			
6	dr			
7	dr			
8	dr			

Equivalent listing
for MICROLOK-
PLUS™

	Out #1	In #1
1	?	dr
2	?	?
3	?	?
4	dr	?
5	dr	
6	dr	
7	dr	
8	dr	
9	?	
10	?	

Trigger List: 10 System Time: 00:00:00:000 Timer List: 0
 Command->INP 1 Program: NBR3 Screen: VO Boards

relay #14 - IN.A input value (0,1 or CR to exit)= 0
 relay #15 - IN.A input value (0,1 or CR to exit)=

4.6.4.9 Relay Set and Clear Commands

A. Non-Timer Relays

The Set (SE __ (CR)) and Clear (CL __ (CR)) commands are used to set or clear all internal and external relays in the source program. Relays can be set or cleared by bit number or name. These commands operate differently, depending on the characteristics of the specific bit. If a bit without a set (clear) delay is set (cleared), the command will produce an immediate change. In the example at the top of the next page, non-timer relays 1 and 2 are set using SE 1 2 (CR) , while non timer-relays 3 and 9 through 11 are cleared with CL 3, 9-11(CR) . (Using the relay names, these commands would be entered as SE OUT.1 OUT.2, and CL OUT.3 OUT.4-OUT.6 (CR))

Note that the changes to the relays are shown in the scroll area. The example shows the last two lines in this particular scroll. All relays that are changed are scrolled on the screen.

-Set and Clear of non-timer relays-

Bit Number - Name	Value	Status	Bit Number - Name	Value	Status
1 OUT.1	set				
2 OUT.2	set				
3 OUT.3	clr				
9 OUT.4	clr				
10 OUT.5	clr				
11 OUT.6	clr				

clear relay #10	OUT.5
clear relay #11	OUT.6

B. Timer Relays

a. General Comments

If a bit is being set and it has a set delay greater than 0, the bit will remain in its current state and be placed on the timer list to be set; this will happen only after the specified time has elapsed. When a bit is placed on the timer list, its set delay will be displayed under the "Status" column. If a bit is invalid and is on the timer queue to be set, and a clear command is issued, the bit will be removed from the timer queue and immediately cleared. If an output bit is not used in any logical equations, then an explicit set or clear command for that bit will cause an error statement, indicating that changing the bit will have no effect on the system.

NOTE: Clearing of a bit also follows from this description.


b. Examples

The example on the next page uses timer relay T1, which has set and clear times both greater than 0.


First, T1 (initially invalid) is cleared (CL 19 (CR) , or CL T1 (CR)). When an invalid timer bit is cleared, the clear time of that relay is recorded in the "Status" column and the relay is placed on the timer queue. However, the relay itself is neither set nor clear. Note that the Timer List entry in the status line changes from 0 to 1. This shows that one timer relay has been placed on the timer queue. Note also that the change to this relay is noted in the scroll area.

T1 is next given a Set command (SE 19 (CR) , or SE T1 (CR)). The effect of this command (at this time) is to remove T1 from the timer queue and set it. T1 is then given another Clear command. With this action, the relay is placed on the timer queue. When the simulation is conducted and 1000 milliseconds elapse, the bit will be removed from the timer queue and the relay will finally be cleared.


Initial clear of relay T1

Bit Number - Name	Value	Status	Bit Number - Name	Value	Status
					
19 T1	?	1000 clr			
Trigger List: 10		System Time: 00:00:00:000		Timer List: 1	
Command-> CL T1		Program: NBR3		Screen: Relays	
put relay #19	T1	oh timer queue			

Set relay T1

Bit Number - Name	Value	Status	Bit Number - Name	Value	Status
					
19 T1	set				
Trigger List: 10		System Time: 00:00:00:000		Timer List: 0	
Command-> SE T1		Program: NBR3		Screen: Relays	
remove relay #19	T1	from timer queue			

Relay T1 set and queued for clearing

Bit Number - Name	Value	Status	Bit Number - Name	Value	Status
					
19 T1	set	1000 clr			
Trigger List: 10		System Time: 00:00:00:000		Timer List: 1	
Command-> CL T1		Program: NBR3		Screen: Relays	
put relay #19	T1	on timer queue			

Some relays may have a clear or set time equal to 0 msec. In the example below, OUT.7 (clear time of 0 msec.) is given a Clear command (CL 12 <CR> , or CL OUT.7 <CR>). Since this relay has no clear delay, the command has the immediate effect of changing OUT.7 from "invalid" to "clr".

-Initial clear of relay OUT.7-

Bit Number - Name	Value	Status	Bit Number - Name	Value	Status
12 OUT.7	clr				
Trigger List: 10 System Time: 00:00:00:000 Timer List: 1 Command-> CL 12 Program: NBR3 Screen: Relays clear relay #12 OUT.7					

To put this relay on a timer queue for its 1000 msec set interval, a Set command (SE 12 <CR> , or SE OUT.7 <CR>) would be entered:

-Relay OUT.7 set-

Bit Number - Name	Value	Status	Bit Number - Name	Value	Status
12 OUT.7	clr	1000 set			
Trigger List: 10 System Time: 00:00:00:000 Timer List: 2 Command-> SE 12 Program: NBR3 Screen: Relays put relay #12 OUT.7 on timer queue					

An initial Set command for OUT.7 (SE 12 (CR) , or SE OUT.7 (CR)) puts the relay on its set queue. However, since the relay started "invalid" and has not yet changed state, the "Value" of OUT.7 remains "invalid".

-Initial set of relay OUT.7-

Bit Number - Name	Value	Status	Bit Number - Name	Value	Status
12 OUT.7	?	1000 set			
Trigger List: 10 System Time: 00:00:00:000 Timer List: 2 Command-> SE 12 Program: NBR3 Screen: Relays put relay #12 OUT.7 on timer queue					

A Clear command removes OUT.7 from its timer queue, and also eliminates the "invalid" state:

-Relay OUT.7 cleared-

Bit Number - Name	Value	Status	Bit Number - Name	Value	Status
12 OUT.7	clr				
Trigger List: 10 System Time: 00:00:00:000 Timer List: 1 Command-> CL 12 Program: NBR3 Screen: Relays put relay #12 OUT.7 on timer queue					

A relay with a set time of 0, such as OUT.8, would be set and cleared from the invalid state in a corresponding manner.

4.6.4.10 Increment Command

The Increment command (INC __ <CR>) allows the system time to be advanced without executing any logic equations. The value entered after this command must be a factor of 10 milliseconds. If a time is specified that is not a factor of 10 milliseconds, it will be converted to the next smaller valid interval. To demonstrate this command, all timer relays in NBR3 have been queued to their respective set and clear times (Note: the following is the RELAY display):

Bit Number - Name	Value	Status	Bit Number - Name	Value	Status
12 OUT.7	clr	1000 set			
13 OUT.8	set	500 clr			
19 T1	set	1000 clr			

Time is then incremented 10 milliseconds (INC 10 <CR>). Note below that the queued times in the "Status" column have all decreased by 10 milliseconds, and that the System Time has advanced by that amount.

-System time incremented by 10 milliseconds-

Bit Number - Name	Value	Status	Bit Number - Name	Value	Status
12 OUT.7	clr	990 set			
13 OUT.8	set	490 clr			
19 T1	set	990 clr			
Trigger List: 10			System Time: 00:00:00:010		
Command-> INC 10			Program: NBR3		
increment time: 10 msec.			Timer List: 3		
			Screen: Relays		

Next, the system time is incremented by 490 milliseconds (INC 490 <CR>), the remaining time needed to clear relay OUT.8. Note (a) the additional time on the system clock, (b) the reduction of time on relays OUT.7 and T1, (c) the state change for OUT.8, and (d) the reduction in the Timer List total:

-System time incremented by 490 milliseconds-

Bit Number - Name	Value	Status	Bit Number - Name	Value	Status
12 OUT.7	clr	500 set			
13 OUT.8	clr				
19 T1	set	500 clr			
Trigger List: 10			System Time: 00:00:00:500		
Command-> INC 490			Program: NBR3		
increment time: 490 msec.			Timer List: 2		
			Screen: Relays		

4.6.4.11 Display Timers Command

The Display Timers (DI TI <CR>) command gives a listing of all relays currently on the timer queue with an active set or clear delay. This command uses the same basic table as the Display Relays command. In the NBR3 sample program, the DI TI <CR> would produce:

-Display timer relays-

Bit Number - Name	Value	Status	Bit Number - Name	Value	Status
12 OUT.7	clr	500 set			
19 T1	set	500 clr			
Trigger List: 10			System Time: 00:00:00:500		
Command-> DI T1			Program: NBR3		
			Timer List: 2		
			Screen: Relays		

4.6.4.12 Execute Command

The Execute command (EX (CR)) executes logic equations without advancing the system time. A number may be entered with this command, specifying the number of trigger list equations to be executed. If a number is not specified, then all equations on the trigger list will be executed. In the example below, program NBR3 is returned to the point with all timer queues on and at their full values. EX 1 (CR) is entered. This command tells the simulator to exercise only the first equation on the Trigger List. Note that relay OUT.7 has been changed from invalid to clear and the Trigger List total is reduced by 1. This indicates that nine Trigger List equations remain to be executed. When the trigger list total reaches 0, no more logic equations are queued for execution.

Some GENISYS and MICROLOK PLUS non-vital programs may be written such that certain individual logic executions do not result in a change of output. Thus, some Execute commands may appear to have no effect when, in fact, they are changing internal bits (displayed in the scroll area).

-Execute one logic equation-

Bit Number - Name	Value	Status	Bit Number - Name	Value	Status
1 OUT.1	clr				
2 OUT.2	?				
3 OUT.3	?				
9 OUT.4	?				
10 OUT.5	?				
11 OUT.6	?				
12 OUT.7	?	1000 set			
13 OUT.8	clr	500 clr			
14 IN.A	set				
15 IN.B	clr				
16 IN.C	clr				
17 IN.D	clr				
18 STICK	set				
19 T1	set	1000 clr			

Trigger List: 9	System Time: 00:00:00:000	Timer List: 0
Command-> EX 1	Program: NBR3	Screen: Relays
clr relay #1	OUT.1	

4.6.4.13 Trace Command

The Trace command performs the same function as the Execute command, however the actual boolean logic statements are displayed in the scroll area, as they are executed. The following example repeats the operation of the Execute command in the previous section, however three logic equations are executed rather than one. With the Trace command, this is done by entering TR 3 (CR) :

-Trace three logic executions-

Bit Number	Name	Value	Status	Bit Number	Name	Value	Status
1	OUT.1	clr					

Trigger List: 8

Command-> tr 3

clear relay #1

LINE 23

<CR> to execute equation

OUT.1

ASSIGN (IN.8 @ IN.A) TO OUT.2;

System Time: 00:00:00:000

Program: NBR3

Timer List: 3

Screen: Relays

Note that (a) three logic equations were executed with this command (Trigger list down to 8), (b) this particular equation which ran through the scroll uses the standard GENISYS shorthand notation for XOR (@), and (c) the prompt that notes this logic equation can be executed by entering a carriage return.

By using the No Display (NO <CR>) command, the Display Relays list will disappear from the screen allowing a larger number of traced logic equations to be viewed at the same time.

4.6.4.14 Run Command

The Run Command (RU __ <CR>) executes logic equations and increments system time. Like the Increment command, this command is entered with a time representing how long the system is to be run. If no time is specified (RU <CR>), the system will execute all equations on the Trigger List. If the timer relay list has any entries, the first relay will be removed from the list and the system time will be incremented. If the program contains no timer relays, the default will execute all equations on the trigger list.

In the example at the top of the next page, NBR3 is once again set with all Trigger List equations waiting to be executed and all timers at their full increments. RU 100 <CR> is entered. Equations will be executed, and the system time incremented until 100 milliseconds has elapsed.

Next, the system is run without a specified time interval. Since the lowest queued timer relay delay is 400 milliseconds (relay OUT.8), the system increments at this value: Note that relay OUT.8 has now cleared and that 500 milliseconds remain on relays OUT.7 and T1.

The No Display command can also be used in conjunction with the Run command to permit a larger scroll area. Also, the Run Command can be operated while observing the Display IO screen, to observe the actual card inputs and outputs

Run System for 100 Milliseconds

Bit Number - Name	Value	Status	Bit Number - Name	Value	Status
1 OUT.1	clr				
2 OUT.2	clr				
3 OUT.3	clr				
9 OUT.4	set				
10 OUT.5	?				
11 OUT.6	set				
12 OUT.7	clr	900 set			
13 OUT.8	set	400 clr			
14 IN.A	clr				
15 IN.B	clr				
16 IN.C	clr				
17 IN.D	clr				
18 STICK	set				
19 T1	set	900 clr			

Trigger List: 0
Command-> RU 100

System Time: 00:00:00:100
Program: NBR3

Timer List: 3
Screen: Relays

set relay #11 OUT.6
set relay #19 OUT.4

increment time 100 msec.

Run System (No Increment Specified)

Bit Number - Name	Value	Status	Bit Number - Name	Value	Status
1 OUT.1	clr				
2 OUT.2	clr				
3 OUT.3	clr				
9 OUT.4	set				
10 OUT.5	?				
11 OUT.6	set				
12 OUT.7	clr	500 set			
13 OUT.8	set				
14 IN.A	clr				
15 IN.B	clr				
16 IN.C	clr				
17 IN.D	clr				
18 STICK	set				
19 T1	set	500 clr			

Trigger List: 0
Command-> RU

System Time: 00:00:00:500
Program: NBR3

Timer List: 2
Screen: Relays

increment time 400 msec.

4.6.4.15 Value Command


The Value command (VA ___ (CR)) may be used to display the value of any desired relays. For example, if a Run command has been carried out with the Display IO screen, the Value command can be used to check back on a relay that already changed state. In the example below, VA 18 19 (CR) (or VA STICK T1 (CR)) would show the states of the internal relays that are not displayed on the screen:

-Value of relays 18 and 19-

Listing for
GENISYS®
Program

	Out #1	Out #2	IN #1	IN #2
1	clr	clr	clr	clr
2	clr	set	clr	clr
3	clr			
4	clr			
5	clr			
6	clr			
7	clr			
8	clr			
<hr/>				
9	set			
10	?			
11	set			
12				
13				
14				
15				
16				

Equivalent
Listing for
MICROLOK PLUS™
Program



	Out #1	In #1
1	clr	clr
2	clr	clr
3	clr	clr
4	clr	
5	clr	
6	clr	
7	clr	
8	clr	
9	set	
10	?	
11	set	
12	clr	
13	set	
14		
15		
16		

Trigger List: 0
Command->VA 18 19

relay #18 STICK set
relay #19 T1 set

System Time: 00:00:01:350
Program: NBR3

Timer List: 2
Screen: I/O Boards

4.6.4.16 Read Command

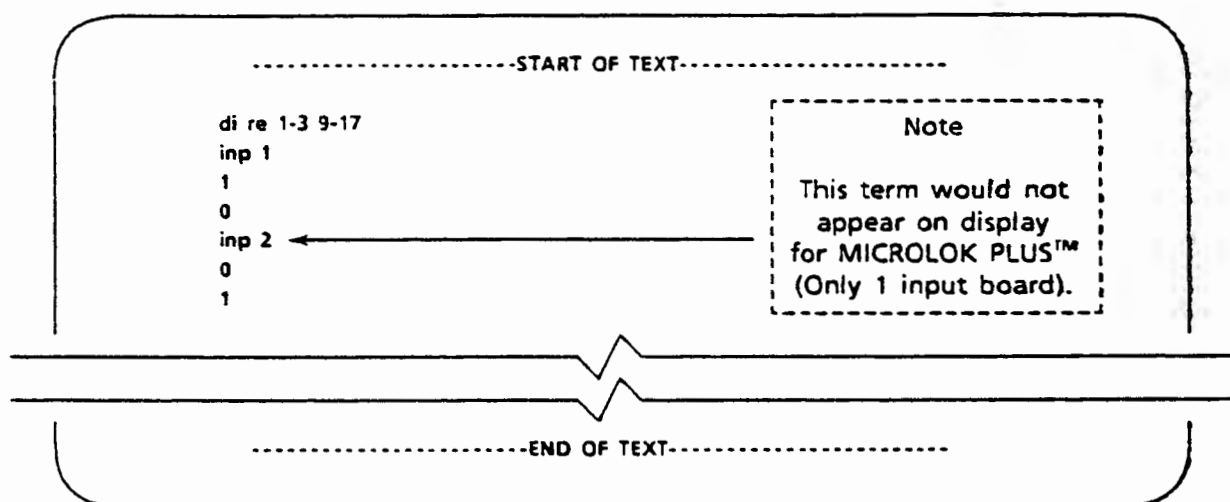
The Read (REA (CR)) command may be used to read commands from a file, rather than the keyboard. It is designed to simplify the re-entering of commands with long lists of items at the beginning of a simulation, or to execute a series of commands in sequence.

When the read command is invoked, the initializing file name is requested. The default extension for this file is .GSI. (If a file extension is entered, it will override this default.) If the command READ NBR3 (CR) is entered, the file used is NBR3.GSI. If the command is simply READ CR , the file name is prompted by INIT FILE-- . The default extension of .GSI is always used by the simulator.

In the example below, the editor is used to create file NBR3.GSI. The display relays and input commands are entered so that (a) only active relays are displayed, and (b) the four inputs on the input board have alternating values:

When the text editor is exited, these commands will be available in the simulator. While running the simulator, enter READ NBR3 <CR> ; the commands will be read from that file and executed.

-Set-up Display Relay and Input commands in file-



Two special commands are valid when reading an Init file, Pause and Continue. To temporarily suspend the reading of commands from an Init file, a Pause command may be entered into the file. When this command is read from the file, the simulator pauses and accepts keyboard commands. At this point, the user may enter other commands. When the continue command is entered, the Read command continues to process commands from the Init file. The Pause command will remain in effect until the Continue command is entered.

4.6.4.17 Print Command

The Print (PR <CR>) command is used to convert a compiler EPROM table (i.e., a .GCD file) back into readable statements. Only the assign statements in the source program are returned. This command can be used to generate the logic equations if a compiler listing is not available.

To execute the Print command with the NBR3 sample program, the user would enter PR NBR3 <CR>. The default file extension for this command is .GEQ. If another extension is desired, it must be entered in the command. When conversion of the EPROM tables is complete, the following message will appear:

File NBR3.GEQ contains the logic equations

To obtain the logic equations themselves, use the Quit command (QU <CR>). To leave the simulator, type NBR3.GEQ <CR>. In this instance, the .GEQ extension must be used.

The format and syntax of displayed assign statements will differ slightly from the statements in the source listing, however they are functionally identical. This is a normal feature of the simulator system. For example, in the NBR3 program, the assign statements are written as follows:

22	ASSIGN IN.A AND IN.B	TO OUT.1;
23	ASSIGN IN.A XOR IN.B	TO OUT.2;
24	ASSIGN IN.A OR IN.B	TO OUT.3;
25	ASSIGN NOT IN.A	TO OUT.4;
26	ASSIGN IN.C OR (STICK AND NOT IN.D)	TO STICK;
27	ASSIGN STICK	TO OUT.6;
28	ASSIGN NOT T1	TO T1;
29	ASSIGN T1	TO OUT.7;
30	ASSIGN NOT T1	TO OUT.8;

In the Print command output, these statements would appear as follows:

Logic equations for Program: NBR3

Line 22

ASSIGN (IN.B * IN.A) TO OUT.1;

Line 23

ASSIGN (IN.B @ IN.A) TO OUT.2;

Line 24

ASSIGN (IN.B + IN.A) TO OUT.3;

Line 25

ASSIGN (\sim IN.A) TO OUT.4;

Line 26

ASSIGN (((\sim IN.D) * STICK) + IN.C) TO STICK;

Line 27

ASSIGN STICK TO OUT.6;

Line 28

ASSIGN (\sim T1) TO T1;

Line 29

ASSIGN T1 TO OUT.7;

Line 30

ASSIGN (\sim T1) TO OUT.8;

4.6.4.18 Reset and Quit Commands

The Reset (RES (CR)) command may be used to reset the simulator. With this command:

- (a) All bits are reset (clear or invalid)
- (b) Trigger and timer lists are reset
- (c) System time returned to 00:00:00:000
- (d) All equations are queued on the trigger list

This is the same procedure used when the simulator is initialized with the program name.

The Quit (QU (CR)) command terminates the simulation and exits the simulator.

4.6.4.19 Color CRT Command

NOTE

The color CRT commands are available with Simulator Version 3.0 and higher. In previous versions, the Simulator was designed to be run on a monochrome video display. It was written using the monochrome characteristics of high and low intensities and underlining. If earlier Simulator versions are used on a PC with a color display, the output screens are displayed with monochrome characteristics, which would be displayed using different colors. Since the earlier version uses underlining to divide the screen into regions, colors do not adequately suffice.

The Version 3.00 Simulator makes use of both monochrome and color characteristics to produce more easily viewed displays. The default mode of the Simulator is monochrome.

When using a color display with Version 3.00, the command "COLOR" must be entered to select color graphics, instead of monochrome graphics. Also, a "MONOCHROME" command is available to revert back to the default mode. In either mode, the same volume of information is displayed.

4.7 G.D.S. - EPROM SIZE ESTIMATES PROGRAM

4.7.1 General

The EPROM Size Estimate program can be used to determine the total number of bytes in the completed source program. This program is useful in estimating the approximate number of EPROMs that will be needed to carry the application program. The GENSIZE help screen provides the memory-usage algorithm for this estimate. (The details of this algorithm are not required for determining the number of application EPROMs; they are provided for general reference.) The terms used in the algorithm are listed below:

TNR	Total number of relays defined
TSP	Total number of spare bits defined
TINPT	Total number of inputs defined (Master, Slave, local)
TRP	Total relays in parallel (i.e., equations with two or more relays receiving the same value)
AC	Average number of contacts in an assignment statement
AO	Average number logical operations per assignment statement
NT	Number of timer relays defined
NM	Number of GENISYS units that will be addressed from Master port
SL	Zero (0) if Slave port is not used, one (1) if Slave port <u>is</u> used.
NIOB	Number of I/O boards
MA	Memory used for assignment statements
TMU	Total memory used

The memory requirement estimation for ASSIGN statement (MA) is given by:

$$MA = ((TNR - TSP - TINPT - TRP) * (4 + 2 (4 * AC) + AO)) + (2 * (TRP + TSP))$$

or more simply:

$$MA = ((TNR - TSP - TINPT - TRP) * (6 + (4 * AC) + AO)) + (2 * (TRP + TSP))$$

The total memory requirement estimation of a system (TMU) is given by:

$$TMU = 16 + MA + (6 * TNR) + (4 * NT) + NIOB + 4 + (9 * SL) + (11 * NM) + 2$$

or more simply:

$$TMU = 22 + MA + (6 * TNR) + (4 * NT) + NIOB + (9 * SL) + (11 * NM)$$

NOTE

The above formulas should be used only to obtain an estimate of memory usage. Exact memory usage can only be determined by actually creating the program, compiling it and examining the EPROM.

4.7.2 Sample Execution

A sample execution of the GENSIZE program is shown below:

How many total relays defined?	(integer) -	800
How many total inputs (master + slave + local)?	(integer) -	200
Number of relays with parallel values?	(integer) -	20
Total number of spares defined?	(integer) -	26
How many timers defined?	(integer) -	69
Will the SLAVE port be used?	(y/n) -	y
How many MASTER addresses?	(integer) -	3
How many local I/O boards?	(integer) -	10
Average number of CONTACTS per assignment statement?	(integer) -	6
Average number of OPERATIONS per assignment?	(integer) -	5
Total Memory usage: 24632. bytes		
or: 24.6K		

NOTE

For a MICROLOK PLUS program, the local I/O boards total is limited to 1 or 2.

Each EPROM can store 8K bytes of application programming. Therefore, the above application would require approximately three EPROMs (24.6 - 8) will be required.

4.8 G.D.S. - EPROM PROGRAMMER

NOTE

This section includes references to the Data I/O Corp. Model 21A and 201 EPROM programmer, which were originally supplied with the GENISYS Development System. These references have been retained for current Model 212 users. The Model 21A was replaced in 1987 with the Data I/O Corp. Model 201 EPROM programmer. The Model 201 was replaced with the Data I/O Corporation Model 212 in 1991. GENISYS Development System Versions 1.02 and higher accommodate Models 21A and 201. Version 3.00 is required for the Model 212.

4.8.1 General

The GENISYS EPROM Programmer (GENPROM) is used to make final checks of the compiler EPROM table, and transfer that table to the Controller board EPROM ICs. This program requires the Data I/O Corporation Model 212 EPROM Programmer. The programmer contains a zero insertion force (ZIF) IC socket for the PROM. US&S recommends EPROM J715029-0409 for the compiler program.

4.3.2 Initial Configuration File - G.D.S Versions 1.04 and Higher

The GENPROM program uses a configuration file to store information regarding the EPROM programmer and EPROMs. This program makes use of an "Environment Table" to determine the location of the configuration file (refer to the appropriate DOS manual for details on the Environment Table). To make this determination, the environment table is searched to find the entry "GDSEEPROM". If found, the value of this entry determines the name and location of the configuration file. For example:

Use the SET Command to specify the location of the configuration file:

```
SET GDSEEPROM = C:\GENISYS\GDSEEPROM.CFG
```

This instructs the GENPROM program that the Configuration File is named "GDSEEPROM.CFG" and that it resides in the directory C:\GENISYS. If the entry is not found, the default is "GDSEEPROM.CFG" in the current directory.

The first time the GENPROM program is run, the configuration file is set up to specify the type of Programmer Unit (Model 21A, 201 or 212) and type of EPROM. Once this file is complete, it does not have to be reentered.

NOTE

Refer to section 4.8.6 if running the following procedure with GENISYS Development System Versions 1.01 through 1.03.

1. The first screen asks for selection of Communication Port 1 or Communication Port 2. This allows use of either Port 1 or Port 2 on the back of the computer. Press the "1" key on the terminal for Port 1, or the "2" key for Port 2.
2. The second screen in the routine asks for selection of either the Model 21A, 201 or 212 programmer. Press the "1" key on the terminal for the model 21A, the "2" key for the 201 or the "3" key for the 212.
3. The next screen asks for selection of the EPROM family/pinout code:

If the Model 21A was selected in step 1, the routine will ask for selection of either the US&S-recommended EPROM (Code 63) or the code for the alternate EPROM. (Refer to the Data I/O Model 21A manual for information on the available EPROM types.)

If the Model 201 or 212 was selected in step 1, the routine will ask for selection of two possible US&S-recommended EPROMs (7933 or 4533), or the code for the alternate EPROM. (Refer to the Data I/O Model 201 or 212 manual for information on the available EPROM types.)

Press the appropriate key for the type of EPROM.

4. If an alternate EPROM is to be used, the screen next asks for the appropriate code of that EPROM. Refer to the EPROM Programmer manual for this code.

5. When the RETURN key is hit to enter the EPROM code, the system is now ready for programming procedures. The screen shows the options (i.e. Port 1 or Port 2) selected to this point. If the options are correct, press "Y" to continue. If not, press "N" to re-enter this information.

4.8.3 Programmer Operation - G.D.S. Versions 1.01 and Higher

NOTE

Refer to section 4.8.7 if running the following procedure using GENISYS Development System Version 1.00.

1. Model 201 or 212 Programmer: Turn on the programmer and check that "SELF TEST OK DATA I/O 201 N" appears on the LCD display. This indicates that the programmer has passed its own start-up diagnostics. Do not make any other adjustments on the programmer.

Model 21A Programmer: Open the small access panel on the front of the Model 21A and set the three left-hand rotary switches to 5, 0, and C, respectively. Do not readjust the three right hand switches. Note: These will be the permanent positions of these switches for all future operations of the programmer. They do not have to be reset each time power is turned on. Then turn on the Model 21A and check that the word "PASS" appears on the ADDRESS portion of the digital display. This indicates the programmer has passed its own start-up diagnostics.

2. Type GENPROM (or genprom) and a carriage return (<CR>) to enter the EPROM programmer routine.
3. The file name of the source program should be entered after the prompt ("Enter the name of the PROM file?"), followed by a <CR>.

NOTES

In Versions 1.04 and higher, the program will continue even if an incorrect file name is entered. The user may retry the file name. Versions 1.04 and higher also allow the user to retry the connection to the programmer without terminating the program.

4. When the file name is entered, the program will indicate how many blank EPROMS are needed to carry the entire program. Type in any key (except "X") as indicated by the prompt at the bottom of the screen.
5. Model 201 or 212 Programmer: As requested on the screen, use the two scroll keys on the Model 201 or 212 so that "RS232 PORT" appears on the LCD display. When this message appears, press the ENTER key on the Model 201 or 212.

Next, use the two scroll keys so that "COMPUTER CONTROL" appears on the LCD display. When this message appears, press the ENTER key on the Model 201 or 212.

6. Model 21A Programmer: Execute the instruction "Enter<SELECT>C<SET>" on the programmer.
7. As instructed on the screen, place a blank EPROM in the programmer. Make certain the pin latching lever is in the up position and insert the EPROM with the notch away from the lever, then close the lever.
8. Press any key on the computer when the EPROM is inserted. The screen will then indicate which EPROM (if more than one are required) will be programmed, the hexadecimal base address of that EPROM, the program file name and several instructions. The base addresses are \$4000, \$6000, \$8000, \$A000 and \$C000.
9. When any key is pressed on the computer, the screen should show the following series of messages:

"Blank check of PROM."
"Downloading program to the PROM programmer."
"Verifying contents of the PROM programmer."
"Programming PROM.....Please Wait."

The program is not loaded directly into the EPROM IC when "any key" is pressed. Instead:

- The EPROM is first checked to make certain it is blank and properly inserted in the programmer socket ("Blank check of PROM").
- Next, the program is transferred to temporary memory in the programmer ("Downloading program....").
- Then, the programmer repeats this procedure to make certain the two match ("Verifying contents....."). This is designed to detect any errors that might have been generated in the tables during the first downloading process.
- Finally, the program is loaded into the EPROM itself, a procedure that typically takes about two minutes.
- Model 201 or 212 programmer: The message "COMPUTER CONTROL" remains on during program downloading, checking and transfer to the EPROM. Also, a period (.) cycles from left to right on the LCD display.

Model 21A Programmer: Address characters cycle. These represent the addresses in the EPROM program as they are delivered.

10. Any hardware problems or errors in the transferred messages will be indicated by any of a variety of fault messages at this time. Messages on the computer are described in section 4.8.4. Refer to the Programmer manuals for their particular error messages.
11. If the program requires two or more EPROMs, the computer will display "PROM # ___ has been programmed" and the check sum for that EPROM. If the installed EPROM is not the last in the set, the computer will repeat the "Press any key" message and the program will go back to step 7.

12. If the installed EPROM is the last in the set, the computer will display several messages indicating that the programmer should be reset and turned off.
13. When the programmer is turned off, pressing any key on the computer will exit the GENPROM program.

4.8.4 Error Messages - G.D.S Versions 1.01 and Higher

NOTE

Refer to section 4.8.8 for computer error messages
in G.D.S. Version 1.00.

The tabulation on the following pages lists computer error messages that may appear in the course of the GENPROM procedure. In most cases, the user should repeat the original procedure as the first means of removing the error.

<u>Message</u>	<u>Cause</u>	<u>Remedy</u>
"PROM programmer failed to accept a command."	EPROM programmer not set up properly.	Check Data I/O manual.
"	Hardware malfunction	Repair or replace EPROM programmer cable, and/or computer.
"The PROM file is incorrect: Invalid format. Recompile the GENISYS program and try again."	Ouput of compiler is not in a valid format because:	
	Bad disk	Recompile on new disk.
	EPROM code file modified after creation by compiler.	Recompile.
"The PROM file is incorrect unexpected end of file".	Same as above	Same as above
"Unable to set PROM type."	The EPROM programmer did not recognize the EPROM type because of a set-up problem.	Verify that the EPROM programmer has been set up correctly.
"The PROM is not properly inserted in the PROM socket."	As indicated	Check that the EPROM is fully inserted in the holder, and that the lever is down.
	Defective EPROM	Replace EPROM.
"The PROM is not blank (Possibly a damaged PROM). Error code from PROM Programmer: ____."	During verification step in download, EPROM programmer returned an error code. Refer to programmer manual for error code meanings.	--
"Unable to read the PROM - Error code from the PROM Programmer: ____."	EPROM could not be read by the programmer, error message returned. Refer to programmer manual for error code meanings.	--
"Unable to set base address of the PROM."	Program too large for the GENISYS or MICRO-LOK PLUS hardware.	Check for possible errors and recompile the source program.

<u>Message</u>	<u>Cause</u>	<u>Remedy</u>
"PROM Programmer rejected the data - Error code from PROM Programmer: ____."	Error found by programmer during download, such as illegal address in code, or unknown record type. Compiler output corrupted by bad disk or attempted alteration in compiler code. Refer to programmer manual for error code meanings.	--
"Transmission error during data transfer."	Communication error between programmer and computer, involving parity, framing, overrun or baud rate.	Check settings of rotary switches on programmer (under keypad), and check installation of EIA cable.
"Data verification error - data conflict between computer and PROM Programmer."	During verification step programmer did not receive same data again. This may be caused by:	
	Communication error	Repeat procedure
	Hardware error	Check EIA cable and programmer.
	Bad data media (computer did not read same data).	Use new disk
"The PROM did not program - it may be damaged. Error code from PROM programmer: ____."	Programmer detected an error during final programming phase; usually indicates bad EPROM. Refer to programmer manual for error code meanings.	--
"Unable to successfully connect with PROM programmer."	Programmer terminated operations after a set number of attempts to create a communications link. Usually caused by bad initialization of programmer.	Check settings of rotary switches on programmer (under keypad).

4.8.5 Communications Interrupt

If the EPROM programmer is interrupted while interactive with the computer (i.e., turned off, reset or EIA cable disconnected), the GENISYS EPROM programmer will cease operation. This is indicated by lock-up of the computer (no response to any commands). To remedy this problem, correct the problem and reset the entire system.

4.8.6 Initial Configuration File - G.D.S Versions 1.00 through 1.03

1. The first screen in the routine asks for selection of either the Model 21A, 201 or 212 programmer. Press the "1" key on the terminal for the model 21A, the "2" key for the 201, or the "3" key for the 212.

2. The next screen asks for selection of the EPROM code:

If the Model 21A was selected in step 1, the routine will ask for selection of either the US&S-recommended EPROM (Code 63) or the code for the alternate EPROM. (Refer to the Data I/O Model 21A manual for information on the available EPROM types.)

If the Model 201 or 212 was selected in step 1, the routine will ask for selection of two possible US&S-recommended EPROMs (7933 or 4533), or the code for the alternate EPROM.

Press the appropriate key for the type of EPROM.

3. If an alternate EPROM is to be used, the screen next asks for the appropriate code of that EPROM. Refer to the EPROM Programmer manual for this code.
4. When the RETURN key is hit to enter the EPROM code, the system is now ready for programming procedures.

4.8.7 Programmer Operation - G.D.S. Versions 1.00

1. Before entering the GENPROM program, make certain the compiler debug switch (for the simulator) is turned "off". This may be done by changing the switch to $\$D-\backslash$, or erasing the switch (switch-off default in effect). If this is not done, a series of error messages will appear at the start of the GENPROM program.
2. Open the small access panel on the front of the Model 21A and set the three left-hand rotary switches to 5, 0, and C, respectively. Do not readjust the three right hand switches. Note: These will be the permanent positions of these switches for all future operations of the programmer. They do not have to be reset each time power is turned on.
3. Turn on the Model 21A and check that the word "PASS" appears on the ADDRESS portion of the digital display. This indicates the programmer has passed its own start-up diagnostics.

4. Type GENPROM (or genprom) and a carriage return (<CR>) to enter the EPROM programmer routine. This will generate a cover screen that remains throughout the program. This screen shows the version of the EPROM programmer.
5. The file name of the source program should be entered after the prompt ("Enter the name of the PROM file?"), followed by a <CR> .
6. When the file name is entered, the program will indicate how many blank EPROMs are needed to carry the entire program. Type in any key (except X) as indicated by the prompt at the bottom of the screen.
7. To place a blank EPROM in the programmer, make certain the pin latching lever is in the up position and insert the EPROM with the notch away from the lever.
8. Next, execute the instruction "Enter <SELECT> C <SET> on the EPROM Programmer", which appears at the bottom of the screen.
9. The screen will then indicate which EPROM (if more than one are required) to insert in the programmer, the hexadecimal base address of that EPROM, the program file name and several instructions. The base addresses are \$4000, \$6000, \$8000 \$A000 and \$C000.
10. When "any key" is pressed, the screen should show the following series of messages in succession:

Blank check of PROM

Downloading program to the PROM programmer

Verifying contents of the PROM programmer

Programming PROM.....Please Wait

The program is not loaded directly into the EPROM IC when "any key" is pressed. First, the EPROM is checked to make certain it is blank and properly inserted in the programmer socket ("Blank check of PROM"). Next, the program is transferred to temporary memory in the programmer ("Downloading program...."). Then, the programmer repeats this procedure to make certain the two match ("Verifying contents....."). This is designed to detect any errors that might have been generated in the tables during the first downloading process. Finally, the program is loaded into the EPROM itself, a procedure that typically takes about two minutes. (Note during the initial EPROM downloading, download check and final loading that the ADDRESS characters on the Model 21A are cycling. These present the addresses in the EPROM program as they are delivered.)

11. Any physical problems in the hardware, or errors in the transferred messages will be indicated by any of a variety of fault messages at this time. These are described in section 2.8.8

12. When EPROM programming has been successfully completed, the message "FROM #__ has been programmed" will appear, and the check sum for that EPROM will be displayed. The check sum may be used to distinguish the EPROMs.
13. If more than one EPROM is needed to hold the tables, the "any key" message will appear. When such a key is pressed, the system will go back to step 8. Otherwise, the terminal will display several messages indicating that the programmer should be reset and turned off.
14. When the programmer has been turned off, pressing any key will exit GENPROM.

4.8.8 Error Messages - G.D.S Version 1.00

Computer error messages available with GENISYS Development System Version 1.00 include all of those under later Versions, plus the following message:

<u>Message</u>	<u>Cause</u>	<u>Remedy</u>
"The Program was compiled with the DEBUG switch ON, it will not execute on the GENISYS hardware. The program must be recompiled without the DEBUG option."	As indicated	Recompile the program without the debug switch (switch-off default) or reset the switch to ASD-

4.8.9 EPROM Programmer Driver - Color Display

The Version 3.00 GENISYS EPROM Programmer Driver program supports color video displays. In previous versions, the program would only produce displays with black and white characters. The new version makes use of colors, and also supports the Data I/O Corp. Model 212 EPROM programmer.

SECTION V
MISCELLANEOUS PROGRAM DESIGN NOTES - GENISYS AND MICROLOK PLUS

5.1 LOGIC AND TIMING OVERFLOWS

If the application program causes the system to attempt to place too many elements on logic or timer queues, a system overflow will occur. This will NOT be recognized by the executive software as a fault. The excess elements will not be placed in a queue and the expected event will not happen. There will be no indication that the logic equation or timing delay was not processed.

When there is no logic, timing or I/O to process, the executive software will systematically execute all the logic equations through an idle loop. If the idle loop executes an equation and discovers that a bit value is in error, the correct value will be assigned to the bit. In this way, logic equations and timing delays that were not performed as a result of logic or timer queue overflows may be executed at a later time. Depending on the size of an application and system loading, this time will vary.

5.2 TIMING ELEMENTS

5.2.1 Introduction

Any output or internal bit in an application program can have a timing delay. The delay emulates slow pick-up and slow drop-away relays. Individual timing delays range from 10 milliseconds to 25 minutes. Each bit can have a set delay (slow pick), a clear delay (slow drop) or both. This time is the delay between when the logic equation is executed and when the value of the bit is changed.

5.2.2 General Processing

After a logic equation is executed, each bit (assigned the result of the equation) is evaluated to see if its value will change because of the result of the logic equation. If a bit gets a new value, the system checks whether there is a timer delay on that bit before executing the change of state. If there is a delay, information is placed on a timer queue along with the length of the delay. When the delay expires, a new value is assigned to the bit. The change of state does not take effect until the time delay has expired. If a change in system status occurs that causes the bit to retain its original value before the delay expires, the time delay is cancelled and the bit never changes state.

5.2.3 Parameters

Time delays are specified in milliseconds, seconds or minutes. Timing accuracy depends on the units selected. The Executive software has three internal timing queues for different time ranges. The timing of a GENISYS or MICROLOK PLUS system is based on a 10 millisecond processor interrupt. Every 10 milliseconds, the timer chip on the processor board generates an interrupt. The interrupt routine then schedules the timer queue handler.

Shorter time delays are handled at this level every 10 milliseconds. These time delays are defined in milliseconds and are within the range of 10 to 2500 milliseconds.

The next level of timer accuracy is the 100 millisecond timer queue. Every 10th time the 10 millisecond queue is scheduled, a 100 millisecond queue is also serviced. This queue handles those time delays, specified in seconds, within the range of 1 to 25 seconds.

The last timer queue is based upon a 6 second time frame. This queue is scheduled once every 60 times that a 100 millisecond queue is serviced. The queue handles all time delays in excess of 25 seconds up to the maximum delay of 25 minutes.

Each timer queue has an internal timing accuracy that is one half the base time. These are as follows:

<u>Timer Queue Range</u>	<u>Accuracy</u>
10 millisecond	+/- 5 milliseconds
100 millisecond	+/- 50 milliseconds
6 second	+/- 3 seconds

5.2.4 Skew Time

The internal timing accuracy of each timer queue is the same with all GENISYS and MICROLOK PLUS non-vital systems. However, the executive software is also subject to a 45 millisecond skew time. The skew time is related to system loading and varies within a given unit over time, depending on the present loading conditions. In smaller systems, there is little or no skew time. In a larger and more complex system, there may be a skew during high load times. High load times occur primarily when serial data is processed. A high baud rate or a large number of serial data bits contributes most heavily to a system skew factor. The system skew time, added to internal timer queue accuracies, gives these worst case system timing accuracies:

<u>Timer Queue Range</u>	<u>Accuracy</u>
10 millisecond	+/- 50 milliseconds
100 millisecond	+/- 95 milliseconds
6 second queue	+/- 3.45 seconds

Because of the skew factor, US&S recommends that all time delays be defined at greater than 100 milliseconds. This will insure that the desired delay is preserved. The restriction does not have to be observed if the time delay is used to alter the order in which logic equations are queued. For queuing order, smaller time delays can be used. The two-queue option will still cause "breaks" equations to be executed before "makes" equations, even if the "makes" are queued first. Therefore, a longer time delay may be necessary to force a "make" equation to occur before a "break" equation.

5.3 VALIDATION OPTION

5.3.1 Introduction

The Validation Option (Validate Value compiler switch) prevents the GENISYS or MICROLOK PLUS non-vital system from delivering data until all inputs required to determine the output state have been received. This option is primarily intended for reset conditions. If an unexpected reset occurs, outputs should be left in the state they held before the reset until a new, valid state is determined by receiving inputs and processing logic. If the Validation Option is turned off, an output could be performed before all inputs have been received. This may cause a momentary change in outputs until all inputs have been received.

For example, "0" represents an unoccupied track circuit. It is being read as a local input and is assigned as a serial output for transmission back to an office. During the time the track is occupied, the system resets. If the Validation Option is turned on, the local input (showing occupancy) is read before the serial output can be sent to the office. The indication always shows the correct status. If the Validation Option is turned off, the serial output may occur before the local input. As a result, the information sent to the office may temporarily show a clear track.

5.3.2 Parameters

When validation is enabled, all local input bits are considered invalid until the input board has been successfully accessed and the data read. All serial input bits are considered invalid until they are received over the serial port. All other bits are initialized with a validation value of 0.

If a logic equation can be resolved to a valid value, bits assigned to the statement are also marked as valid and assigned the result of the equation. If a final result is not valid, the bits assigned to the statement retain their current value and validation status.

When output processors (local or serial) format output states for delivery, no invalid outputs may be delivered. Since the outputs are not delivered on a bit-by-bit basis, one invalid output may prevent other outputs from being delivered, whether or not they are valid. On the local output boards, all 16 bits are delivered simultaneously. If any one of these bits is invalid, the entire board is prevented from delivering outputs. On the serial ports, any one invalid bit will prohibit an entire 8 bit byte from being delivered.

5.3.3 Recommendations

If the Validation Option is turned on, check the list file generated by the development system compiler to verify there are no "unassigned" outputs on the serial links or the local boards.

When upgrading a GENISYS or MICROLOK PLUS non-vital system to a higher revision of the executive PROM IC29 and using the Validation Option ("on"), make sure the new configuration does not cause operational problems.

With revision 7 and higher of the executive software, the Validation Option resolves more logic equations than earlier revisions. As a result, local or serial outputs may be delivered that would not have been delivered with the earlier software revisions.

The following tabulation lists the applications of the Validation Option with the different executive software revisions:

<u>Statement</u>			<u>Revision No.</u>	<u>Result</u>
VALID	OR	VALID	0 and higher	VALID
VALID	XOR	VALID	0 and higher	VALID
VALID	AND	VALID	0 and higher	VALID
	NOT	VALID	0 and higher	VALID
INVALID	OR	INVALID	0 and higher	INVALID
INVALID	XOR	INVALID	0 and higher	INVALID
INVALID	AND	INVALID	0 and higher	INVALID
	NOT	INVALID	0 and higher	INVALID
VALID	OR	INVALID	0 - 6	INVALID
VALID	OR	INVALID	7 and higher	? (*See below)
VALID	XOR	INVALID	0 and higher	INVALID
VALID	AND	INVALID	0 - 6	INVALID
VALID	AND	INVALID	7 and higher	? (*See below)
0	OR	INVALID	7 and higher	INVALID
1	OR	INVALID	7 and higher	VALID VALUE OF 1
0	AND	INVALID	7 and higher	VALID VALUE OF 0
1	AND	INVALID	7 and higher	INVALID

When the Validation Option is used ("on") in executive software Revisions 7 and higher, there are two identities that allow partial processing of equations with invalid parameters:

$$\begin{array}{rcl}
 * & 1 & \text{OR } X \quad 1 \\
 & 0 & \text{AND } X \quad 0
 \end{array}$$

These identities are not evaluated in Revisions 0 through 6. Any invalid parameter in Revisions 0 through 6 causes an invalid result.

With Revision 7 and higher, the value of "X" in the above identities is irrelevant, even if it is invalid. In Revisions 7 and higher, the identities may be applied to process logic with invalid parameters. This is particularly useful where there is both a remote control and a local control for the same function: `ASSIGN ((RC AND RI) OR (LC AND LI)) TO OUTPUT`

With Executive software Revisions 0 through 6, this function will not work in Local Control (LC) unless the Remote Control (RC) function is valid. This could cause problems in cases of code line failure. The function will work with Revisions 7 and higher.

5.4 LOGIC QUEUING AND EXECUTION

5.4.1 Comparison of Hardware and Software Relay Logic

There are limits to which the GENISYS or MICROLOK PLUS non-vital systems can emulate actions and reactions of a non-vital relay system. Relay systems, which are based on electrical hardware connections, processes multiple logic functions in parallel. GENISYS and MICROLOK PLUS, which are based on software and a microprocessor, process multiple logic functions sequentially. For example, where a single contact in a relay system is used in processing two different logic functions, both logic functions start processing simultaneously when the relay changes state. In GENISYS and MICROLOK PLUS, however, one equation is processed before the other.

Also, when a relay changes state in an actual relay system, there is a very brief time when neither the front or back contact has energy applied. When a "relay" changes state in the GENISYS or MICROLOK PLUS non-vital logic program, there is no true transfer time.

In GENISYS and MICROLOK PLUS, those equations that use the energized state of a bit can be equated to the front contacts of a relay. Those equations that use a de-energized state can be equated to the back contacts of a relay. An example is provided in Figure 5-1:

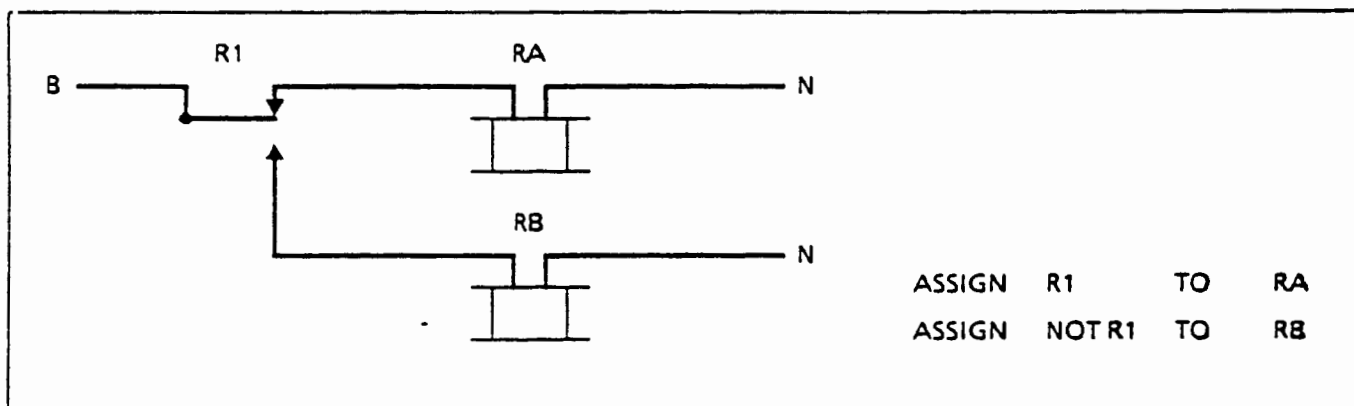


Figure 5-1. Example of Front and Back Contact Assignments

In this example, the RA equation is the front contact and the RB equation is the back contact. All equations that use the NOT operand are a back contact. In a relay system, there will be a measurable time, when R1 starts to drop, when neither the front nor back contacts has energy applied. In GENISYS and MICROLOK PLUS, this transfer is instantaneous. Either the front or back contact has energy applied at all time.

5.4.2 Breaks Before Makes Rule

The Executive software recognizes that the order in which equations are executed affects the internal and output states of the system. To emulate relay circuits operation as closely as possible, the executive employs the traditional "break before make" rule of relay systems. It determines which equations involve the "front contact" and which equations involve the "back contacts" of a relay in a typical application. Depending on whether the relay

In Figure 5-2, the following occurs: R1 is dropped. R2, R3, R4 and R5 are picked. RA and RB are picked while RC is dropped. If R2 drops, RC may or may not pick, depending on the queuing option chosen.

One-queue option: When R2 drops, both the RA and RC equations are placed on the queue to be executed. RA, which involves the break, is queued and executed first. When the RA equation is executed, RA drops. This causes the RB equation to be queued. Since there is only one queue, the RB equation is placed in the queue after the RC equation. The RC equation is then removed and executed. RB is still picked and R2 is dropped, causing RC to pick. Since RC has now changed state and is used in the RC equation, this equation goes back on the queue after the RB equation. The RB equation is removed from the queue and executed. This causes RB to drop. The RC equation is removed and re-executed. RB is now down. However, since RC has already picked and has a valid path through a stick circuit, RC remains picked.

Two-queue option: When R2 drops, both the RA and RC equations are queued to be executed. The RA equation, which involves a break, is placed on the break queue. The RC equation, which involves a make, is placed on a make queue. Any equation queued because of a break is done before those caused by a make. Therefore, the RA equation is removed and executed first. When the RA equation is executed, RA drops. This causes the RB equation to be queued. Since the RB equation is being queued because of a break in the RA relay, the RB equation is placed on the break queue. Since this equation is in the break queue, it is removed and executed before the RC equation, which is in the make queue. This occurs even though the RC equation was queued prior to the RB equation. When the RB equation is executed, RB drops. The RC equation is now removed and executed. At this time, RB has already dropped, therefore RC will not pick.

Differences in queuing options can be masked by using timers. If RA or RB had a clear delay (emulating a slow drop relay), RC will always pick. If RC is not to pick, it could be given a set delay emulating a slow pick relay. Either configuration may achieve the desired result.

SECTION VI
MISCELLANEOUS APPLICATION INFORMATION - GENISYS AND MICROLOK PLUS

6.1 LOCAL I/O

6.1.1 Using Slave Units as I/O Processors

A GENISYS unit is capable of handling a maximum of 256 local input and output bits. A MICROLOK PLUS unit non-vital section is capable of handling a maximum of 32 local input and output bits. Certain applications, such as those with local control panel or event recorder, may have local I/O requirements for a single location that exceed the above bit limit. Extreme care should be used when attempting to handle the additional I/O with Slave GENISYS units. Where possible, the logic and local I/O should be divided between units to share the system load as much as possible. If only one GENISYS unit is designated to handle all logic and timing functions, as well as local and serial I/O, the chances of a system overload are increased.

6.1.2 Determining the Control Delivery Time

6.1.2.1 Introduction

The Control Delivery Time is the time that the system holds a deliver pulse on each relay-output board. This time is selected with a compiler switch in the application program or on the Controller PCB. For the Control Delivery PCB N451441-3601 and the Control and Delivery PCB N451441-4701, the Control Delivery Time represents the time the outputs remain energized. With the Constant Delivery PCB N451441-7101 (which has a final stick output), outputs remain energized after the Control Delivery Time expires. Refer to section 4.2.2.3 for the Control Delivery Time compiler switch options in the application program. Refer to section 8.4.2 for the Control Delivery Time switch settings on the Controller PCB.

6.1.2.2 Selection Considerations

The optimum Control Delivery time is determined by the application. The selected time must be long enough to activate the devices connected to the GENISYS or MICROLOK PLUS non-vital outputs, but not longer than required. The Executive software delivers controls to one relay-output board at a time. When changes are to be delivered to several boards at once, the changes are sent to these boards one at a time. The deliver pulse must be held on each output board for the length of the Control Deliver Time.

If the control Deliver Time is set too long, it can delay an output sequence for a GENISYS system. For example, a GENISYS system has 10 relay-output boards and each of the 10 boards needs controls delivered to it. If the Control Deliver Time is set to 1 second, there will be a 10 second lag between delivery of controls to the first board and delivery of controls to the last board, assuming a change on each board. If the external devices require each of the GENISYS outputs to be energized for 1 second, the 10 second output lag cannot be avoided. However, if devices can tolerate a shorter Control Delivery Time, this time should be utilized.

US&S suggests a 70 millisecond Control Delivery Time for the Constant Delivery PCB N451441-7101. This time is sufficient to energize the particular type of relays used on this board.

For the Control Delivery (N451441-3601) and Control and Delivery (N451441-4701) PCBs, the Control Delivery Time must be long enough for the slowest acting device to recognize and accept the output state.

6.2 SERIAL COMMUNICATIONS TIMING

There is no standard formula for determining the proper timing elements of a GENISYS or MICROLOK PLUS serial data link. When requested by the customer, US&S performs these calculations on a project-by-project basis, using the unique specifications of each customer's system. Contact the US&S Headquarters office for assistance on the serial timing calculations.

SECTION VII
SERIAL COMMUNICATIONS PROTOCOL - GENISYS AND MICROLOK PLUS

7.1 INTRODUCTION

NOTE

On the following pages the character '\$' preceding a number denotes a hexadecimal (base 16) number. The \$ sign is not a transmitted character.

7.1.1 Message Format

The GENISYS or MICROLOK PLUS non-vital serial communications protocol consists of the following sequence of messages:

CONTROL CHARACTER (HEADER)	STATION ADDRESS	DATA BYTE(S)	SECURITY CHECKSUM (CRC)	TERMINATION CHARACTER (TRAILER)
----------------------------------	--------------------	--------------	-------------------------------	---------------------------------------

1. Control Character

- a. Each message must start with a control or header character, which denotes the type of message.
- b. Different control characters are defined for the Master and Slave ports.

2. Station Address

- a. If the transmitting system is a Master unit, this address refers to the location of the receiving Slave unit.
- b. When the Slave unit receives the transmission, the station address is decoded. If the message applies to that station, the message is processed and acted upon. Messages addressed to other Slave units are ignored.
- c. If the transmitting system is Slave unit, this address refers to the address of that Slave unit.
- d. The Master unit expects a message from the Slave unit it initially addressed, including the address of that Slave unit.

3. Data Byte

- a. Data is always sent as two 8-bit bytes, including a byte number followed by a data byte.
- b. Each byte is identified with a byte number which starts at zero. For example, a message with 24 data bits would be packaged as follows:

<u>Byte No.</u>	<u>Data Bits</u>
0	1-8
1	9-16
2	17-24

- c. With this format, it is not necessary to send the entire data base to transmit selected bits.

4. Security Checksum

- a. This is the automatic, 2-byte cyclical redundancy check (CRC) checksum.
- b. When this checksum is included, all other message bytes (except the termination character) are included in the checksum.

5. Termination Character

The message is always terminated by the termination (or "trailer") character \$F6.

7.1.2 Message Sequence

The following discussion is based on the use of a GENISYS or MICROLOK PLUS unit as the non-vital serial link Master. If some other type of computer or the US&S Programmable RCCI is being used as a Master unit, it may not be possible to implement this sequence. Also, the discussion does not refer to systems with a check-back function. The check-back function can only be used in certain types of system configurations, and affects system timing. Contact US&S for information on the use of the check-back function in a GENISYS or MICROLOK PLUS non-vital system.

In a normal communication sequence, the Master unit sends a message to a particular Slave unit. The Slave unit decodes this message, then formats and transmits a response. Once the Master unit receives and decodes the message from the Slave unit, the Master unit determines if another message is to be sent to that same Slave unit, or continue to the next station in the polling cycle. In either case, the Master always formats and transmits a new message. The appropriate Slave unit responds and the cycle continues.

The tabulation at the top of the next page shows a background polling cycle from a GENISYS or MICROLOK PLUS Master for a system with seven field stations.

The first two lines indicate the sequence for system initialization. The remaining lines shows a normal background polling scheme used by a Master unit. This table only shows the first control character to be sent by the Master and does not show any field responses or additional messages sent by a Master to the same field.

Station No.						
S1	S2	S3	S4	S5	S6	S7
Messages						
\$FD	\$FD	\$FD	\$FD	\$FD	\$FD	\$FD
\$FC	\$FC	\$FC	\$FC	\$FC	\$FC	\$FC
\$FD	\$FB	\$FB	\$FB	\$FB	\$FB	\$FB
\$FC	\$FB	\$FB	\$FB	\$FB	\$FB	\$FB
\$FB	\$FB	\$FB	\$FB	\$FB	\$FB	\$FD
\$FB	\$FB	\$FB	\$FB	\$FB	\$FD	\$FC
\$FB	\$FB	\$FB	\$FB	\$FD	\$FC	\$FB
\$FB	\$FB	\$FB	\$FD	\$FC	\$FB	\$FB
\$FB	\$FB	\$FD	\$FC	\$FB	\$FB	\$FB
\$FB	\$FD	\$FC	\$FB	\$FB	\$FB	\$FB
\$FD	\$FC	\$FB	\$FB	\$FB	\$FB	\$FB
\$FC	\$FB	\$FB	\$FB	\$FB	\$FB	\$FB
\$FB	\$FB	\$FB	\$FB	\$FB	\$FB	\$FD

The Master unit will start a communication with a Slave unit using any one of the following messages, and the Slave unit will respond as indicated:

Master Unit Message

Slave Unit Response

\$FD

\$F2 (All data being sent)

\$FC

\$F2 (Changed data being sent)

or

\$F1 (No data changes to report)

\$FB

\$F2 (Changed data being sent)

or

\$F1 (No data changes to report)

When the Master unit decodes the Slave unit's response, the Master unit will format its next message:

- a. If the Slave unit responds with an \$F1 message, the Master unit may transmit a message to the next Slave unit in the polling cycle.
- b. If the Slave unit responds with an \$F2 message, the Master unit will send an acknowledgement message (\$FA) before continuing with the next Slave unit.
- c. In turn, the Slave unit responds to this acknowledgement with an \$F1 or \$F2 message.
- d. If the Slave unit sends another \$F2 message, the Master unit sends another acknowledgement until the Slave unit responds with an \$F1 message:

<u>Master Unit Message</u>	<u>Slave Unit Response</u>
\$FC or \$FB	\$F2 (All data being sent)
	or
\$FC or \$FD	\$F2 (Changed data being sent)

The Master unit then sends an \$FA message in response to the Slave unit's \$F2 message:

<u>Master Unit Message</u>	<u>Slave Unit Response</u>
\$FA	\$F2 (Changed data being sent)
	or
	\$F1 (No data changes to report)

As long as the Slave unit sends new information in \$F2 messages to the Master unit, the Master unit will continue to acknowledge the received data. When the Slave unit terminates the communication with an \$F1 message, the Master can continue the polling cycle with the next Slave unit.

7.1.3 Good and Bad Messages

A "good" message conforms to the format described in the previous section and passes the CRC check, when applicable. A "bad" message may include an invalid header (control character), an incorrect station address, a bad CRC checksum (where applicable) or an invalid or missing termination character.

A Slave unit only responds to a good message that includes its correct station address:

- a. A message start is regarded as a valid control character. If a valid control character is noted, but only a partial message is received, the message receiving process is aborted.
- b. Any current or pending transmit messages at that Slave unit are also aborted. However, data changes and latched bits are not lost. Only the transmission is aborted.

The receive port on the Slave unit is always enabled. The Slave unit cannot block a transmission from the Master unit, except when the receive port circuit is disabled because of an absent DCD signal.

The Master unit is programmed to expect a response to all messages transmitted to the Slave unit(s).

- a. Messages should not be received from the Slave units when no transmission has been made from the Master unit.
- b. When the Master unit is transmitting a message, its receive port circuit is disabled.
- c. When the full message is transmitted, the No Response Time Out timer is started. This timer represents that maximum amount of time allowed between send and receive transmissions (refer also to section 4.2.2.3).
- d. The No Response Time Out is stopped when the starting control character is received from the Slave unit. A second time-out is then started. This time-out is the maximum time that can exist between two received characters. It is set at 300 milliseconds. Each time a character is received, this time-out is reset until the \$F6 character is seen.
- e. When the termination (trailer) character is received, the message is then processed by the Master unit logic.
- f. If the message is from a Slave station other than the one originally addressed (wrong station address), has an invalid format or does not pass the CRC check, it is considered an invalid message by the Master unit logic.
- g. If an invalid message is received or either of the receive data time-outs elapses, the Master unit logic ignores any partially received data and goes to a special "bad-receive" handler.

The first time Slave unit fails to respond properly to a Master unit transmission:

- a. The Master unit will retransmit the same message once. This is done to allow for intermittent line noise that may have interfered with the first transmission.
- b. If this second attempt also fails, the intended Slave unit is designated as off-line in the Master unit logic.
- c. The SLAVE.ON.xx bit is cleared in the application logic and the Master continues the polling cycle with the next Slave unit.
- d. When a Slave unit is designated as off-line, future polling cycles will only consist of one transmission from the Master unit (no retransmission occurs) until that Slave unit responds to the initial transmission. (Note: This function is only available with Executive software (IC29) Revisions 3 and higher.)

7.2 DETAILED DESCRIPTION

7.2.1 General Specifications

The default GENISYS and MICROLOK PLUS non-vital data transmissions consist of 10 bits: 1 start bit, 8 data bits and 1 stop bit (no parity). This can be changed on Controller PCB switch SW7; refer to section 8.4.8. The general format is as follows:

1. The transmission is a modified binary with unique control characters in the range F0 to FE (hex).
2. Character FF is illegal since this is commonly created on noisy lines.
3. Data in the range 00 to EF (hex) is sent as is. Data in the range F0 to FF (hex) is sent as the escape character F0 (hex), followed by the low nibble of the data. For example, a data byte F3 would be sent as F0 followed by 03. The receiver of a message will always "OR" the byte, following an F0 with it and treat the result as one data byte. This refers to all non-control character information (station address, byte number, byte data and CRC-16).
4. Data security will be in the form of CRC-16. The generator polynomial will be the standard CRC-16 polynomial $X(16) + X(15) + X(2) + 1$.
5. Communication status bits (MASTER.ON and SLAVE.ON.xx) will be provided which are accessible through the high-level language. These bits will be set upon receipt of a valid message addressed to that unit. If no message is received within the (selectable) time-out period, the bits will be modified.
6. Control and indication byte addresses 223-255 (\$E0-\$FF) are reserved for status type information currently defined in these bytes. These include the following:

Control byte address \$E0 (sent in \$FC or \$F9 message):

bit 0 - Data base complete	4 - Future
(0 = data base not complete)	5 - Future
1 - Use check-back controls	6 - Future
2 - Use secure poll only	7 - Future
3 - Allow common command	

Indication byte address \$E0 (sent in \$F2 message):

bit 0 - Data base complete	4 - Future
(0 = need controls)	5 - Future
1 - Use check-back controls	6 - Future
2 - Use secure poll only	7 - Future
3 - Allow common-control command	

The Message formats supported include the following:

1. Header byte, station address, terminator - three byte format used for messages which do not require security.
2. Header byte, station address, message, CRC-16, terminator - variable length format for messages with checksum.

NOTE

On the following pages the character '\$' preceding a number denotes a hexadecimal (base 16) number.

7.2.2 Master to Slave Data Transmission

7.2.2.1 Poll Command

A Poll Command is sent to allow a Slave unit to respond. The Slave responds with F1, F2.

Non-secure format:

<u>Bytes</u>	<u>Description</u>	<u>Comments</u>
1	\$FB	(Poll command)
1	\$01-\$FF	(station number (0 not used))
1	\$F6	(terminator)

Secure Format:

<u>Byte</u>	<u>Description</u>	<u>Comments</u>
1	\$FB	(Poll command)
1	\$01-\$FF	(station number (0 not used))
1	\$00-\$FF	(low byte of CRC-16)
1	\$00-\$FF	(high byte of CRC-16)
1	\$F6	(terminator)

7.2.2.2 Acknowledge Data Command

The Acknowledge Data Command is sent to acknowledge data from a Slave unit. The Slave responds with F1, F2.

<u>Byte</u>	<u>Description</u>	<u>Comments</u>
1	\$FA	(Acknowledge Data command)
1	\$01-\$FF	(station number (0 not used))
1	\$00-\$FF	(low byte of CRC-16)
1	\$00-\$FF	(high byte of CRC-16)
1	\$F6	(terminator)

7.2.2.3 Control Command

The Control Command is used to send controls to a station. With checkback controls, the slave responds with F3 message. With non-checkback controls, the Slave responds with F1, F2.

<u>Byte</u>	<u>Description</u>	<u>Comments</u>
1	\$FC	(Control Command)
1	\$01-\$FF	(station number (0 not used))
1	\$00-\$FF	(control byte number)
1	\$00-\$FF	(control byte)

Repeat control byte number and data as required.

1	\$00-\$FF	(low byte of CRC-16)
1	\$00-\$FF	(high byte of CRC-16)
1	\$F6	(terminator)

7.2.2.4 Recall Indications Command

The Recall Indications Command is used to recall all indications. The Slave responds with an F2 message.

<u>Byte</u>	<u>Description</u>	<u>Comments</u>
1	\$FD	(Recall Indications command)
1	\$01-\$FF	(station number (0 not used))
1	\$00-\$FF	(low byte of CRC-16)
1	\$00-\$FF	(high byte of CRC-16)
1	\$F6	(terminator)

7.2.2.5 Execute Controls Command

The Executive Controls Command is the response to a valid control checkback. The Slave responds with F1, F2.

<u>Byte</u>	<u>Description</u>	<u>Comments</u>
1	\$FE	(Execute Controls)
1	\$01-\$FF	(station number (0 not used))
1	\$00-\$FF	(low byte of CRC-16)
1	\$00-\$FF	(high byte of CRC-16)
1	\$F6	(terminator)

7.2.2.6 Common Control Mode

The Common Control Mode is used to deliver control byte 0 or 1 to one or all Slave units. Slave does not respond. Control bytes 0 and 1 are always accessible by the control (\$FC) command regardless of the state of the "Allow Common Command" bit in the mode byte. When accessed via the \$FC command, a response is given as is normally the case with controls.

<u>Byte</u>	<u>Description</u>	<u>Comments</u>
1	\$F9	(Common Control command)
1	\$00-\$FF	(station number 0 = all stations)
1	\$00-\$01	(control byte nbr (0,1))
1	\$00-\$FF	(common control byte)
Repeat control byte number, control byte.		
1	\$00-\$FF	(low byte of CRC-16)
1	\$00-\$FF	(high byte of CRC-16)
1	\$F6	(terminator)

7.2.3 Slave to Master Data Transmission

NOTE

If the Slave unit does not properly decode the Master unit message, the Slave will not respond.

7.2.3.1 Acknowledge Master Response

The Acknowledge Master Response is sent as response to messages when no other messages are pending.

<u>Byte</u>	<u>Description</u>	<u>Comments</u>
1	\$F1	(Acknowledge master response)
1	\$01-\$FF	(station number (0 not used))
1	\$F6	(terminator)

7.2.3.2 Indication Data Response

The Indication Data Response is used to send indication data to the Master unit.

<u>Byte</u>	<u>Description</u>	<u>Comments</u>
1	\$F2	(Indication data response)
1	\$01-\$FF	(station number (0 not used))
1	\$00-\$FF	(indication byte number)
1	\$00-\$FF	(indication byte)
Repeat byte number and data as required.		
1	\$00-\$FF	(low byte of CRC-16)
1	\$00-\$FF	(high byte of CRC-16)
1	\$F6	(terminator)

7.2.3.3 Control Checkback Command

The Control Checkback Command is used to verify controls from the Master unit if the Checkback Control mode is set. The Master responds with FE message. If the following message to this station is not a valid FE message for this station, the control is lost.

<u>Byte</u>	<u>Description</u>	<u>Comments</u>
1	\$F3	(Control Checkback command)
1	\$01-\$FF	(station number (0 not used))
1	\$00-\$FF	(control byte number)
1	\$00-\$FF	(control byte)
Repeat control byte number and data as required.		
1	\$00-\$FF	(low byte of CRC-16)
1	\$00-\$FF	(high byte of CRC-16)
1	\$F6	(terminator)

7.2.4 Control Code Summary

<u>Code</u>	<u>Message</u>	<u>Format</u>	<u>Affected by Modes?</u>
(Slave to Master)			
\$F1	Acknowledge Master	Non-Secure	no
\$F2	Indication Data	Secure	no
\$F3	Control Checkback	Secure	yes (1)
(End of Text)			
\$F6	End of Text character	--	--
(Master to Slave)			
\$F7	(future)	--	--
\$F8	(future)	--	--
\$F9	Common Controls	Secure	yes (2)
\$FA	Ack. Indication & Poll	Secure	no
\$FB	Poll	Either	yes (3)
\$FC	Controls	Secure	no
\$FD	Recall	Secure	no
\$FE	Execute	Secure	yes (4)

- (1) If Checkback Controls are not enabled in the \$E0 byte, Control Checkback is not a valid protocol character.
- (2) If Common Controls are not enabled in the \$E0 byte, Common Control is not a valid protocol character.
- (3) Default is secure poll, non-secure poll can be selected in the \$E0 byte.
- (4) If Checkback Controls are not enabled in the \$E0 byte, execute is not a valid protocol character.

SECTION VIII
SUPPLEMENTAL DATA

8.1 TOKEN AND PARSING ERROR/WARNING MESSAGES

The following tabulation lists all parsing error and warning messages that may appear while developing the GENISYS or MICROLOK PLUS non-vital program on the compiler. With parsing errors, the compiler will point with carets () to the area where it first detected a problem, and stop. The carets may actually be pointing to an item past the location where the actual error occurred. The programmer should look for the error at or before the carets.

Parsing Error		Token Error	
Ref. No.	Type of Error	Ref. No.	Type of Error
1.	PROGRAM statement missing	1	Input line truncated.
3.	INTERFACE statement missing	2	Too many Identifiers declared.
4.	Remote I/O specification expected	3	Numeric constants greater than four digits.
5.	Address specification expected	5	ID: identifier contains an illegal character.
6.	INPUT or OUTPUT word specification expected	6	Word more than 12 characters
7.	Relay name expected	7	Unknown compiler switch.
8.	Unexpected ID found after ", "	8	Missing or zero value for compiler switch.
10.	Incorrect interface format		
11.	COMMA or COLON expected		
12.	Semicolon or comma expected		
13.	Invalid "SET=OPTION"		
14.	Invalid timer units specified		
15,16	Invalid timer "SET/CLEAR" statement		
17.	Missing clear parameter on timer declaration		
18,19	Invalid timer declaration		
20.	Invalid declaration format		
21.	"BEGIN" missing		
22.	Missing ASSIGN statement		
23.	ASSIGN statement or end of program expected		
24.	Invalid expression syntax		
	(* warning*)		
55.	Semicolon missing		

8.2 SEMANTIC ERROR MESSAGES

Following is a list of semantic error messages that may appear while developing the GENISYS or MICROLOK PLUS non-vital program on the compiler:

1: More than (n) ID's in ID list

ID lists have various limits corresponding to the limits imposed by the hardware. For example, an INPUT WORD or OUTPUT WORD statement in the LOCAL I/O section may only contain 16 relay names for GENISYS and two relay names for MICROLOK PLUS, due to the physical limit on each of the input or output boards. Above, (n) specifies the limit that was exceeded.

2: Set or clear delay more than 25 minutes

The pick-up or drop delay was specified as greater than 25 minutes. If longer times are desired, assign one time-delayed relay to another to obtain the desired net effect.

3: Multiple-defined relay: (relay name)

A user-defined relay name was specified in more than one I/O statement or VAR statement for internal bits. Any bit that appears in two or more definitions is illegal.

4: ID '(relay name)' Multiple-defined set/clear delay

The same relay was specified in two TIMER statements. Each relay may have only one timing specification.

5: ID '(relay name)' Input Bit assigned as Timer Bit

Input bits get their values externally, and hence cannot have user specified pick-up or drop delays.

6: Invalid switch: (s)

The user has specified a switch (%\$...) incorrectly.

7: LOCAL I/O section already defined

The user has attempted to define two LOCAL I/O sections.

8: MASTER I/O section already defined

The user has attempted to define two MASTER sections.

9: ID '(relay name)' INVALID ASSIGNMENT TO A PHYSICAL INPUT BIT

The (relay name) specified in the ASSIGN statement is an input bit on the bit on the LOCAL I/O. It is illegal to assign these bits values with ASSIGN statements.

10: SLAVE I/O section already defined

The user has attempted to define a second SLAVE section.

11: More than one address in SLAVE I/O definition

The user has attempted to define multiple address specifications in the Slave section. The GENISYS unit can be programmed to respond to only one Slave address.

12: ID '(relay name)' should be Relay name

The user has used the identifier defined as the program name where a relay name should have been used.

13: Master Station address already defined

The user has already used this station address in a previous definition.

14: Illegal Master Station Address

The user has defined a slave unit with an illegal address to be in communication with this unit's MASTER port. Each SLAVE must have an address specified in the range: 1-255.

16: Illegal SLAVE station address

The user has defined an invalid address for the Slave port of this unit. The address of the SLAVE port must have an address specified in the range: 0-255. If zero is specified, the value of the hardware switch 5 will be used by the run-time system to determine what address the SLAVE port will respond.

18: All output bits must be specified before input bits

The user has defined OUTPUT information after specifying the INPUT specifications for a given section. In every I/O section, the OUTPUT's must be specified first.

19: Use of SPARE in ASSIGNMENT statement

The special identifier "SPARE" has been used in an ASSIGN statement. Only user-defined relay names may be used.

20: ID: '(relay name)' is undefined'

The user has specified an undefined relay name in a TIMER statement or an ASSIGN statement. Every relay used in these statements must be defined in some I/O section or the VAR section.

21: Invalid Switch setting. Switch: (s)

The setting specified for this switch is invalid. Check individual switch documentation.

22: ID: '(relay name)' multiple-assigned

The (relay name) shown appears in two ASSIGN statements. Each relay may only be ASSIGNED in one (1) statement.

23: ID: '(relay name)' already specified in this ID list

The (relay name) shown appears twice in an ID list.

25: Assignment of SPARE as timer bit

SPARE may not appear in a TIMER statement.

26: Assignment of SPARE as internal bit

SPARE may not appear in a VAR statement.

27: ID '(relay name)' Set and clear delay for this timer bit are both 0.

A timer bit was defined to have both a set delay and clear delay of zero. The system will work properly, but additional processing time will be required for the bit shown.

28. Program name: "name" contains illegal characters.

Refer to sections 4.2.1.3 and 4.2.4.1 for allowed characters.

8.3 CODE SYSTEM PRE-PROGRAMMED EPROMS

US&S provides a series of EPROMs which contain a program for configuring the GENISYS or MICROLOK PLUS non-vital section as a CTC system field code unit. These programs are used to interface the local I/O with the SLAVE port. In these programs, the serial input is mapped to the local outputs, and the local inputs are mapped to the serial outputs. The following list shows the default values used for compilation:

<u>Function</u>	<u>Default Value</u>
Slave Baud Rate	Hardware Switch*
Master Baud Rate	1200 BPS
Control Delivery Time	Hardware Switch*
Security	Automatic**
Validity Check	On
Two Queue Option	Off

*Setting on the Controller PCB switch.

**Default for this switch in G.D.S. Version 1.00 is called "Off".

The Code System EPROMs are tabulated in Table 8-1. The "Controls" and "Indications" listings represent the maximum number of output and input PCBs, respectively, that may be installed in the GENISYS or MICROLOK PLUS cardfile for these functions. EPROMs may be selected for applications which do not use the precise control/indication configurations listed in the table, provided

the precise number of output boards does not exceed the listed maximum in the table. For example, a GENISYS system requiring five relay-output and five optical-input boards could use EPROM N451575-0910. The empty indication board slots in this case are ignored by the program. The N451575-0910 code system EPROM is the GENISYS standard.

NOTES

Any of the following EPROMS may be used in the non-vital section of MICROLOK PLUS since this unit is limited to two non-vital I/O slots.

The Code System EPROMs contain complete programs that cannot be modified to include custom user logic. If the GENISYS or MICROLOK PLUS unit is to include features in addition to a simple code system, it must be custom programmed.

Table 8-1. Code System Application EPROMs

Part Number	I/O PCB Configurations		Part Number	I/O PCB Configurations	
	Controls	Indications		Controls	Indications
N451575-0904	0	16	N451575-0913	9	7
▪ -0905	1	15	▪ -0914	10	6
▪ -0906	2	14	▪ -0915	11	5
▪ -0907	3	13	▪ -0916	12	4
▪ -0908	4	12	▪ -0917	13	3
▪ -0909	5	11	▪ -0918	14	2
▪ -0910	6	10	▪ -0919	15	1
▪ -0911	7	9	▪ -0920	16	0
▪ -0912	8	8			

8.4 CONTROLLER PCB HARDWARE PROGRAMMING (See Figure 8-1)

8.4.1 Slave Port Baud Rate (SW1)

Rotary switch SW1 sets the baud rate of the Slave Port on the Controller PCB. This rate may also be selected in the application program (refer to section 4.2.2.3). The Master port baud rate is only selected on the application program. If no Slave port baud rate is specified in the program, the system will operate at the rate selected on SW1. If the Slave port baud rate is selected in the program (compiler switch %\$S1 through %\$SF), this rate will override the SW1 setting. Table 8-2 on page 8-7 lists the available Slave Port baud rates on SW1.

NOTE

In Executive software revisions 11 and higher, a setting of "0" on the Master port baud rate (in the application logic program) causes the system to default to the hardware baud rate for the Slave port.

Figure 8-1. Controller PCB Manually Selected Options

GENISYS[®]

(NON-VITAL LOGIC EMULATOR)

MICROLOK PLUS[™]

(VITAL + NON-VITAL CONTROL PACKAGE)

CONTROLLER PCB

BOARD: N451441-56 REV. NO. _____

EXEC. SOFTWARE, IC29: REV. NO. _____

(IC29 PART NO. N451575-0901)

SERIAL NO. _____

LOCATION _____

OPTIONS SELECTED

SLAVE ADDRESS
TYPE H*: __ S*: __
VALUE _____

KEY ON _____

KEY OFF _____

CONTROL DELIVERY TIME
TYPE H*: __ S*: __
VALUE _____

SLAVE BAUD RATE
TYPE H*: __ S*: __
VALUE _____

COMMUNICATION (EIA OR TTL) _____

JUMPERS 1-10 SET A-B FOR EIA
 B-C FOR TTL

*HARDWARE-SELECTED
**SOFTWARE-SELECTED

CARRIER TEST AND ADJUST

ROCKERS (X = EITHER POS. OK)

1	2	3	4	TEST FUNCTION
0	0	X	X	NORMAL OPERATION
1	0	X	X	SLAVE PORT TEST ONLY
0	1	X	X	MASTER PORT TEST ONLY
1	1	X	X	MASTER AND SLAVE PORT TESTS
X	X	1	X	50% DUTY CYCLE TRANSMITTED ON SERIAL LINE
X	X	0	0	W/SW4 OUT: ALL SPACES TRANSMITTED
X	X	0	1	W/SW4 OUT: ALL MARKS TRANSMITTED
X	X	0	0	W/SW4 IN: ALL MARKS TRANSMITTED
X	X	0	1	W/SW4 IN: ALL SPACES TRANSMITTED

NOTE
REFER TO TEXT FOR SET-UP OF THESE SWITCHES.

KEY-ON DELAY

ROCKERS

1	2	3	4	DELAY
0	0	0	0	ZERO
1	0	0	0	4
0	1	0	0	8
1	1	0	0	12
0	0	1	0	16
1	0	1	0	20
0	1	1	0	24
1	1	1	0	28
0	0	0	1	32
1	0	0	1	36
0	1	0	1	40
1	1	0	1	44
0	0	1	1	48
1	0	1	1	52
0	1	1	1	56
1	1	1	1	60

KEY-OFF DELAY

ROCKERS

5	6	7	8	DELAY
0	0	0	0	ZERO
1	0	0	0	4
0	1	0	0	8
1	1	0	0	12
0	0	1	0	16
1	0	1	0	20
0	1	1	0	24
1	1	1	0	28
0	0	0	1	32
1	0	0	1	36
0	1	0	1	40
1	1	0	1	44
0	0	1	1	48
1	0	1	1	52
0	1	1	1	56
1	1	1	1	60

CONTROL DELIVERY TIME

0 = 10 mSEC
1 = 30 mSEC
2 = 70 mSEC
3 = 130 mSEC
4 = 250 mSEC
5 = 1 SEC
6 = 2 SEC
7 = 4 SEC

SERIAL PORT DATA BYTE FORMAT
(REVISION 7 AND HIGHER OF EXEC. PROM IC29, ONLY)

CLOSED: ODD PARITY
OPEN: EVEN PARITY

CLOSED: PARITY DISABLED
OPEN: PARITY ENABLED

CLOSED: 1 STOP BIT
OPEN: 2 STOP BITS

SLAVE STATION ADDRESS

1 = 1 5 = 16
2 = 2 6 = 32
3 = 4 7 = 64
4 = 8 8 = 128

KEY-ON DELAY

KEY-OFF DELAY

SLAVE BAUD RATE

0 = 150 4 = 2,400
1 = 300 5 = 4,800
2 = 600 6 = 9,600
3 = 1,200

JUMPERS 16 - 210

ABC ABC ABC ABC ABC
000 000 000 000 000

SET ALL JUMPERS TO A-B POS. FOR RS-423

ABC ABC ABC ABC ABC
000 000 000 000 000

SET ALL JUMPERS TO B-C POS. FOR TTL

JUMPERS 11 - 15

COMMUNICATIONS MODE

NOTE

DIP SWITCHES SW5, SW6, AND SW7 MOUNTED REVERSE OF SWITCHES SW4, SW5, SW6 AND SW7 ON MICROLOK CODE SYSTEM PCB

CARRIER TEST AND ADJUST (SEE ALSO SW7)

CARRIER MODE

SYSTEM WIDE RESET

PUSH DOWN ROCKER TO "OPEN" SIDE FOR "1"

PUSH DOWN ROCKER TO NUMBERED SIDE FOR "0"

NOT USED

OPEN

OPEN

OPEN

CARRIER

OPERATE

LED 1 MTRD

LED 2 MNTS

LED 3 MNRD

LED 4 MOCN

LED 5 STAD

LED 6 SRTS

LED 7 SRAD

LED 8 SOKD

LED 9 WATCHDOG

LED 10 DELVER

Table 8-2. Hardware-Defined Slave Port Baud Rates (SW1)

<u>SW1 Setting</u>	<u>Baud Rate</u>	<u>SW1 Setting</u>	<u>Baud Rate</u>
0	150	4	2400
1	300	5	4800
2	600	6	9600
3	1200		

8.4.2 Control Delivery Time (SW2)

Rotary switch SW2 sets the Control Delivery Time for the local relay-output PCBs. This rate may also be selected in the application program (refer to section 4.2.2.3). If no Control Delivery Time is specified in the program, the delay will be determined by the setting of SW2. If the Control Delivery Time is selected in the program (compiler switch %01 to %0F), this rate will override the SW1 setting. Table 8-3 lists the available Slave Port Control Delivery Times on SW2.

NOTE

US&S recommends against using the 10 msec. setting. Some relays may fail to react to this short of a control pulse.

Table 8-3. Hardware-Defined Control Delivery Times (SW2)

<u>Control Del. SW2 Setting</u>	<u>Delay</u>	<u>Control Del. SW2 Setting</u>	<u>Delay</u>
0	10 msec.	4	250 msec.
1	30 msec.	5	1 sec.
2	70 msec.	6	2 sec.
3	130 msec.	7	4 sec.

8.4.3 Carrier Mode (SW3)

Toggle switch SW3 is set according to the application. When placed in the CARRIER position, the constant carrier mode is invoked and the RTS line on the serial port remains high throughout the data transmission. Constant carrier can only be used on the Master port; the Slave port always uses the key delays. When this switch is placed in the OPERATE position, the Key-On and Key-Off Delays are activated. The RTS line on the serial port is toggled high for the specified number of bit times before the a byte is sent, and toggled low for the specified number of bit times after the byte has been sent across the serial line. This feature is applied when using modems which require toggling of the RTS line a certain number of times before the data byte is sent.

NOTES

With Executive EPROM (IC29) Revision 3 and higher, the incoming carrier (DCD) on the Master and Slave ports must be turned off and kept off before a outgoing transmission can be made. Otherwise, transmissions from these ports will be aborted. This applies to half-duplex communications only.

When the outgoing carrier from the Master port is on (RTS on), changes in the incoming carrier will not affect the data transmission. When the outgoing carrier from the Slave port is on, changes in the incoming carrier will cause the transmission to be aborted. These conditions apply to half-duplex communications only, and are in effect with all revisions of the Executive EPROM.

8.4.4 Slave Station Address (SW5)

DIP switch SW5 selects the station address of a GENISYS or MICROLOK PLUS non-vital Slave station. (This switch may be ignored when the unit is a first Master or stand-alone unit.) If no station address is selected in the application program (address "0" defined), the station address will be defined by the value on switch SW5. The station address can be any value in the range of 0 to 255. Table 8-4 lists rocker positions and corresponding address values. A "0" indicates the rocker is closed and "1" indicates the rocker is open. Selected bits are added together to form the desired address. For example, Slave station address 3 is created by placing rockers 1 and 2 to the "1" position and all remaining rockers to the "0" position.

Table 8-4. Slave Unit Station Address

<u>SW5 Rocker</u>	<u>Bit Value</u>	<u>SW5 Rocker</u>	<u>Bit Value</u>
1	1	5	16
2	2	6	32
3	4	7	64
4	8	8	128

8.4.5 Key-On and Key-Off Delays (SW6)

DIP switch SW6 selects the carrier Key-On and Key-Off delays. Rockers 1 through 4 define the Key-On delays while rockers 5 through 8 define the Key-Off delays. Table 8-5 lists rocker positions and corresponding delay values. A "0" indicates the rocker is closed and "1" indicates the rocker is open.

Table 8-5. Key-On and Key-Off Delays (SW6)

Switch 6 Rockers					Switch 6 Rockers				
<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>Key-On Delay</u>	<u>5</u>	<u>6</u>	<u>7</u>	<u>8</u>	<u>Key-Off Delay</u>
0	0	0	0	zero delay	0	0	0	0	zero delay
1	0	0	0	4	1	0	0	0	4
0	1	0	0	8	0	1	0	0	8
1	1	0	0	12	1	1	0	0	12
0	0	1	0	16	0	0	1	0	16
1	0	1	0	20	1	0	1	0	20
0	1	1	0	24	0	1	1	0	24
1	1	1	0	28	1	1	1	0	28
0	0	0	1	32	0	0	0	1	32
1	0	0	1	36	1	0	0	1	36
0	1	0	1	40	0	1	0	1	40
1	1	0	1	44	1	1	0	1	44
0	0	1	1	48	0	0	1	1	48
1	0	1	1	52	1	0	1	1	52
0	1	1	1	56	0	1	1	1	56
1	1	1	1	60	1	1	1	1	60

8.4.6 Communications Mode Select (Jumpers)

Jumpers J1 through J10 select TTL or EIA RS-423 (RS-232C compatible) communications for the Master and Slave ports of the GENISYS or MICROLOK PLUS non-vital unit. (If the unit is a stand-alone type, the jumper locations are ignored by the software.) Each jumper position is set up A-B-C with the plug installed either A-B or B-C. Table 8-6 lists jumper positions for the different communications modes on the Master and Slave ports.

NOTE

For applications where high noise immunity is required, RS-423 is recommended.

Table 8-6. Master and Slave Communications Mode Select (TTL or RS-423)

<u>Port</u>	<u>Communications Mode</u>	<u>Jumper No.</u>	<u>Jumper Position</u>
Master	RS-423	J6 - J10	A-B
Slave	RS-423	J1 - J5	A-B
Master	TTL	J6 - J10	B-C
Slave	TTL	J1 - J5	B-C

8.4.7 Serial Port Test (SW7)

Rockers 1 through 4 of DIP switch SW7 are used to test controller PCB serial ports and local modem. To set up the controller for normal operation, set rockers 1 and 2 to "0".

8.4.8 Serial Port Data Byte Format (SW7)

On GENISYS and MICROLOK PLUS systems with Revision 7 and higher of the Executive EPROM (IC29), SW7 rockers 6, 7 and 8 select the data byte format for the serial ports. (On previous revisions, these were set at 1 start bit, 8 data bits, 1 stop bit and no parity). Table 8-7 lists switch settings for the available format options:

Table 8-7. Serial Data Byte Format (SW7)

<u>SW7 Rocker</u>	<u>Position</u>	<u>Format Selected</u>
8	Closed	1 Stop Bit (default)
	Open	2 Stop Bits
7	Closed	Parity Disabled (default)
	Open	Parity Enabled
6	Closed	Odd Parity
	Open	Even Parity

To simulate earlier revisions (pre Revision 6), rockers #7 and #8 should be placed in the closed position. Rocker #6 is ignored by the system when rocker #7 is closed.

Rockers #7 and #8 are mutually exclusive. Either 2 stop bits may be used, or parity may be enabled. If parity is enabled, (rocker #7 open), then only 1 stop bit will be used (rocker #8 will be ignored by the system).

UNION SWITCH & SIGNAL 

A member of the ANSALDO Group
5800 Corporate Drive Pittsburgh, PA 15237

SERVICE MANUAL 6300A

APPENDIX A
PARTS LIST - DEVELOPMENT SYSTEM

GENISYS[®]
NON-VITAL LOGIC EMULATOR

MICROLOK PLUS[™]
VITAL + NON-VITAL CONTROL PACKAGE
(NON-VITAL SECTION)

Up to and including:
Executive Software Revision 11
Application Logic Software Version 3.0

October, 1991
A-10/91-2645-1
ID0312F, 0313F

COPYRIGHT 1991, UNION SWITCH & SIGNAL INC.
PRINTED IN USA

ANSALDO
Trasporti

DEVELOPMENT SYSTEM (G.D.S.) EQUIPMENT

<u>Item</u>	<u>Description</u>	<u>US&S Part No.</u>
EPROM Programmer	Data I/O Corp. Model 212	J703105-0003
Cable	EPROM Programmer to PC (25-Pin)	N451458-7201
EPROM Eraser	Spectronics PE-14T	J703105-0005
Diskette w/Software	G.D.S., Hard and Floppy Disk Versions - 5-1/4"	N451232-0101
Diskette w/Software	G.D.S., Hard and Floppy Disk Versions - 3-1/2"	N451232-0112
Blank Diskette	5-1/4"	J703105-0004
Blank Diskette	3-1/2"	J703105-0008

*Model 201 replaced Model 21A in 1987.
Model 212 replaced Model 201 in 1991.

