UNION SWITCH & SIGNAL

645 Russell Street
Batesburg, SC 29006

SERVICE MANUAL 6700A

APPLICATION LOGIC PROGRAMMING

GENISYS 2000

NON-VITAL LOGIC EMULATOR

August 1996

An ANSALDO Affiliated Company

I INTRODUCTION 1-1

1.1 FAMILY OF MANUALS 1-1
1.2 APPLICATION AND EXECUTIVE SOFTWARE 1-1
1.3 COMPONENTS 1-2
1.3.1 Cardfile 1-2
1.3.2 Controller PBC 1-2
1.3.3 Relay-Output PCBs 1-2
1.3.4 Optical Input PCBs 1-3
1.3.5 Power Supply Converter PCBs 1-4
1.4 PROGRAMMING SPECIFICATIONS 1-4

II COMPONENT DESCRIPTIONS 2-1

2.1 INTRODUCTION 2-1
2.1.1 Programming Language Terms 2-1
2.1.2 Reserved Words 2-2
2.1.3 User-Defined Symbols 2-3
2.1.4 Delimiters 2-3
2.1.5 Formats 2-3
2.1.6 Non-Program Comments 2-3
2.1.7 Compiler Switches 2-4
2.2 PROGRAM EXAMPLES 2-5
2.2.1 Local Input/Output 2-5
2.2.2 Internal Relays and Stick Logic 2-6
2.2.3 Timing Relays 2-7
2.2.4 Master/Slave Communications 2-8
2.3 DETAILED STATEMENT DESCRIPTIONS 2-9
2.3.1 Program Statement 2-9
2.3.2 Interface 2-9
2.3.3 Local Section 2-9
2.3.4 Master 2-10
2.3.4.1 ADDRESS Declaration 2-11
2.3.4.2 OUTPUT Declaration 2-12

2.3.4.3 INPUT Declaration 2-12
2.3.5 SERIAL.SLAVE 2-12
2.3.5.1 ADDRESS Declaration 2-13
2.3.5.2 FLAG Declaration 2-13
2.3.5.2.1 DATATRAIN VIII Specific 2-14
2.3.5.2.2 ALLEN BRADLEY DF1 Specific 2-14
2.3.5.3 OUTPUT Declaration 2-14
2.3.5.4 INPUT Declaration 2-14
2.3.6 ATCS.SLAVE or ARES.SLAVE 2-14
2.3.6.1 FLAG Declaration 2-15
2.3.6.2 ADDRESS Declaration 2-16
2.3.6.3 WIU.ADDRESS Declaration 2-16
2.3.6.4 OUTPUT Declaration 2-16
2.3.6.5 INPUT Declaration 2-16
2.3.7 DC.CODE.LINE 2-17
2.3.7.1 FLAG Declaration 2-19
2.3.7.2 ADDRESS Declaration 2-19
2.3.7.3 INDICATION Declaration 2-19
2.3.7.4 Indication LENGTH Declaration 2-19
2.3.7.5 Indication ADDRESS Declaration 2-19
2.3.7.6 OUTPUT Declaration 2-19
2.3.7.7 CONTROL Declaration 2-20
2.3.7.8 Control LENGTH Declaration 2-20
2.3.7.9 Control ADDRESS Declaration 2-20
2.3.7.10 INPUT Declaration 2-20
2.4 CONFIGURATION ITEMS FOR PROGRAM PARAMETERS 2-20
2.4.1 VAR 2-44
2.4.2 TIMER 2-44
2.4.3 Main Program Body 2-45
2.4.4 ASSIGN Statement 2-45
2.4.5 Run Time System Description 2-46
2.4.6 Logic Processing 2-47

2.4.7 Remote Communications- Predefined Relays 2-48
2.4.8 Valid Bit Option 2-49
2.5 VALIDATION OPTION 2-49
2.5.1 Parameters 2-49
2.5.2 Recommendations 2-50
2.6 LOGIC QUEUING AND EXECUTION 2-50
2.6.1 Comparison of Hardware and Software Relay Logic 2-50
2.6.2 Breaks Before Makes Rule 2-50
2.6.3 Queuing Options 2-51
2.6.4 Relay Models and Programming Techniques 2-52
2.7 GENISYS 2000 DEVELOPMENT SYSTEM (G.D.S.2.) 2-53
2.8 G.D.S.2. - AVAILABLE FILE 2-54
2.9 G.D.S.2. - COMPILER 2-54
2.10 G.D.S.2. - SIMULATOR 2-55
2.10.1 Access to Simulator 2-56
2.10.2 Procedure 2-56
2.10.3 Standard Formats 2-57
2.10.4 Simulator Operation 2-57
2.10.5 Sample Program 2-58
2.10.6 Help Screen 2-60
2.10.7 Display IO Command 2-61
2.10.8 Display Triggers Command 2-61
2.10.9 Display Relays Command 2-62
2.10.10 Remove Command 2-63
2.10.11 Input Command 2-64
2.11 RELAY SET AND CLEAR COMMANDS 2-65
2.11.1 Non-Timer Relays 2-65
2.11.2 Timer Relays 2-65
2.11.3 Increment Command 2-68
2.11.4 Display Timers Command 2-69
2.11.5 Execute Command 2-69
2.11.6 Trace Command 2-70
2.11.7 Run Command 2-71
2.11.8 Value Command 2-73
2.11.9 Read Command 2-74

2.11.10 Print Command 2-74
2.11.11 Reset and Quit Commands 2-76
2.11.12 Color CRT Commands 2-76

III PROGRAMMING PROCEDURES 3-1

3.1 LOGIC AND TIMING OVERFLOWS 3-1
3.1.1 Timing Elements 3-1
3.1.2 General Processing 3-1

IV MISCELLANEOUS PROGRAM DESIGN 4-1

4.1 LOCAL INPUT/OUTPUT(I/O) CONSIDERATION 4-1
4.1.1 Using Slave Units as I/O Processors 4-1
4.1.2 Determining the Control Delivery Time 4-1
4.1.3 Selection Considerations 4-1
4.1.4 Serial Communications Timing Considerations 4-1

V MISCELLANEOUS APPLICATION INFORMATION

5.1 GENISYS SERIAL COMMUNICATION PROTOCOL 5-1
5.1.1 General Message Format 5-1
5.1.1.1 Control Character 5-1
5.1.1.2 Station Address 5-1
5.1.1.3 Data Bytes 5-1
5.1.1.4 Security Checksum 5-2
5.1.2 Basic GENISYS PROTOCOL MASTER to SLAVE MESSAGES 5-2
5.1.2.1 Common Control Message ($F9) 5-2
5.1.2.2 Acknowledge and Poll Message ($FA) 5-2
5.1.2.3 Poll Message ($FB) 5-2
5.1.2.4 Control Data Message ($FC) 5-2
5.1.2.5 Indication Recall (Master Recall) Message ($FD) 5-3
5.1.2.6 Control Execute Message ($FE) 5-3
5.1.3 Basic GENISYS Protocol Slave to Master Messages 5-3

5.1.3.1 Acknowledge Message ($F1) 5-3
5.1.3.2 Indication Data Message ($F2) 5-3
5.1.3.3 Control Checkback Message ($F3) 5-3
5.1.4 GENISYS Master Protocol Drivers Operation 5-3
5.1.5 GENISYS Slave Protocol Driver Operation 5-4
5.1.6 GENISYS Configuration Control Bytes 5-4
5.1.7 Detailed Message Formats for Master to Slave Messages 5-5
5.1.7.1 Common Control Message 5-5
5.1.7.2 Acknowledge and Poll Message 5-5
5.1.7.3 Poll Message 5-5
5.1.7.4 Control Message 5-6
5.1.7.5 Recall Message 5-6
5.1.7.6 Control Execute Message 5-6
5.1.8 Detailed Message Formats for Slave to Master Messages 5-6
5.1.8.1 Acknowledge Message 5-6
5.1.8.2 Indication Data Message 5-7
5.1.8.3 Control Data Checkback Message 5-7
5.1.9 Control Character Summary 5-7

VI SUPPLEMENTAL DATA ________ 6-1

6.1 TOKEN AND PARSING ERROR/WARNING MESSAGES 6-1
6.2 SEMANTIC ERROR MESSAGES 6-2
6.3 CODE SYSTEM PRE-PROGRAMMED EPROMS 6-10
6.4 GENERAL EPROM PROGRAMMING INSTRUCTIONS 6-10
6.4.1 GENISYS 2000 EPROM File Format Information 6-10

Appendix A PARTS LIST

A.1 Development System Equipment A-1

List Of Illustrations

Figure 2-1 ASSIGN Operators and Order of Precedence Samples ... 2-46
Figure 2-2 Queuing Options Reference Diagram ... 2-47
Figure 2-3 Example of Front and Back Contract Assignments ... 2-50
Figure 2-4 Queuing Option Example ... 2-51
Figure 2-5 Conceptual Relay Models for GENISYS 2000 Programming ... 2-52
Figure 2-6 Development System Block Diagram ... 2-53
Figure 2-7 Initial NBR3 Trigger Screen ... 2-62

List Of Tables

Table 2-1 Character Set ... 2-1
Table 2-2 GENISYS 200 Reserved Words ... 2-2
Table 2-3 Delimiters ... 2-3
Table 2-4 Logical Operators Symbols ... 2-45
Table 2-5 ASSIGN Operators Truth Tables ... 2-46
Table 2-6 Trigger List Development Table ... 2-47

This manual provides instructions for programming the non-vital application software of the GENISYS® 2000 Non-vital Logic Emulator (NVLE).

1.1 FAMILY OF MANUALS

This manual is one of two manuals that cover the GENISYS 2000.

- **(SM 6700A)** Non-Vital Logic Emulator Application Logic Programming
- **(SM 6700B)** Installation and Field Maintenance Hardware Installation, Local and Serial Data Interfacing, Field Troubleshooting

The GENISYS 2000 Non-vital Logic Emulator (NVLE) is a general-purpose logic emulator and communication unit that performs 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, that takes inputs, performs 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 centralized traffic control (CTC) system field station, and processing of local indications for a local wayside control panel.

The basic hardware elements of GENISYS 2000 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. Five optional output PCBs are equipped with either 16 single-pole relay outputs or 32 optically isolated solid-state outputs for various relay or control panel lamp driving applications or both. Two optional input PCBs are equipped with 16 or 32 optical-isolator ICs for interfacing of contact-generated input signals. Power supply converter options are tailored for dc voltages from 9.5 to 35 Vdc and standard 120 Vac commercial power.

GENISYS 2000 may be interfaced directly or via many digital communication systems commonly used in remote service. A Master/Slave protocol is used for communications between a computer and GENISYS 2000 unit(s) or between two or more GENISYS 2000 units. Configurations may include stand-alone units with no serial communications links and Master/Slave systems with serial communications links. Master/Slave systems may incorporate one or more Slave GENISYS 2000 units.

Communications electrical interface options include EIA RS-423 (RS-232C compatible) or RS-422 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 2000 unit may control up to 512 inputs and outputs in any combination. Up to 64 Slave units may be controlled by a Master unit, but the typical practical limit is 40 to 50 units.

1.2 APPLICATION AND EXECUTIVE SOFTWARE

The GENISYS 2000 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 railway signal engineer. The finished program is converted into a form that can be entered or “burned” onto two (2) EPROM chips.

The GENISYS 2000 may be programmed by using the GENISYS 2000 Development System (G.D.S.2.). This system enables the user to design and test their own program, and then load it into the system hardware. The G.D.S.2. consists of a personal computer, an EPROM Programmer and the GENISYS 2000 Development System software, which is contained on a single diskette. Presently there are five Executive Software sets. These allow the GENISYS 2000 to interface with different slave protocols. (ATCS, ARES, GENISYS/5XX Field Codes, MCS, and WB&S S2).

1.3 COMPONENTS

1.3.1 Cardfile

The GENISYS 2000 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 left-hand slot, a controller PCB in the second slot and between zero and 16 input and output PCBs in the remaining slots. When output PCBs are present, they must be grouped in the leftmost input/output card slots. All input PCBs must be in card slots to the right of all output PCBs. Empty slots between the last output PCB and the first input PCB must be identified in the application program as spare card slots. Because of these limitations, a GENISYS 2000 cardfile may contain a mix of 16 and 32 bit input and output cards. If required by a specific application, a cardfile may contain all input or all output PCBs in the far right-hand 16 card slots.

1.3.2 Controller PCB

The Controller PCB performs all logical decisions and calculations for the GENISYS 2000 system, and serves as the remote communications interface for any external devices. Primary functions include management of local I/O (via cardfile interface boards) and remote I/0 (via serial data line) as defined by the custom-designed application program. The basic integrity of the controller is continuously verified by the execution of internal watchdog and testing routines.

1.3.3 Relay-Output PCBs

US&S provides five different relay-output PCBs which may be used in the GENISYS 2000 cardfile.

PCB Name	Part Number	Characteristics
16 Bit Control Delivery	N451441-3601	Variable duration pulse delivery (using relay outputs)
16 Bit Constant Delivery	N451441-7101	Continuous delivery (using relay outputs
16 Bit Control and Delivery	N451441-4701	Variable duration pulse delivery duration pulse delivery for use with external polar magnetic-stick storage relays (using relay outputs)
32 Bit Output Control	N451441-9601	Fixed duration pulse or constant delivery (using current sinking solid-state drivers)
32 Bit Output Control (Source)	N451441-9801	Fixed duration pulse or constant delivery (using solid-state current source drivers)

Solid-state output boards with current sinking drivers switch the negative side of the load, while current source drivers switch the positive side of the load. Output board selection is based on the requirements of the application.

1.3.4 Optical Input PCBs

US&S provides two different optical input PCBs which may be used in the GENISYS 2000 cardfile.

16 Bit Optical Input	N451441-7202	Wide input voltage range (AC or DC)
32 Bit Input	N451441-9701	Wide input voltage range (DC only)

Older N451441-5802 optical input boards may also be used.

1.3.5 Power Supply Converter PCBs

US&S provides two power supply converter PCBs for GENISYS 2000 which output operating power for other cardfile PCBs, and the carrier modem interface (+/- 12 Vdc) when required by application. They are as follows:

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

1.4 PROGRAMMING SPECIFICATIONS

Total Bits:	4000 bits maximum (can be divided between local I/0, serial I/O and internal)
Local I/O Boards:	16 maximum per cardfile (any combination)
Local I/O Bits:	512 using 32-bit input and output PCBs
Master-to-Slave Communications:	64 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
Master Baud:	300, 600, 1200, 2400, 4800, 9600, 19200
Slave Baud:	75*, 300, 600, 1200, 2400, 4800, 9600, 19200 *MCS (only)
Active Timing Elements:	300 maximum active at any one time in application logic
Logic Equations Triggered:	4000 maximum

2.1 INTRODUCTION

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. Logical bits are defined in input, output and internal declaration statements and associated with physical inputs and outputs by position. Bits can represent local parallel I/O directly connected to the unit or remote I/O which passes to devices through either the Master or Slave Serial ports. Timing values for either defining set (pick-up) or clear (drop-away) or both delays can be associated with internal or output bits. Boolean statements describe the logical relationships between bits which comprise the system logic.

The complete program, including bit definition and Boolean logic statements is referred to as the source program. The source program is processed by the compiler and converted into data tables which can be read and executed by the Executive Software on the GENISYS 2000 controller board. These data tables are programmed into two EPROMs which are installed on the controller board. The following sections describe how these basic programming operations are accomplished using the GENISYS 2000 Development System.

2.1.1 Programming Language Terms

The Character Set consists of the full ASCII character set, as defined for the user's computer. These are listed below in Table 2-1. Only certain characters may be used to make up user-defined symbols (refer to User-Defined Symbols, on the following page). 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.

FOR USER-DEFINED SYMBOLS	SPECIAL CHARACTERS	SPECIAL CHARACTERS
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 **__**	Closed Parens. **)**	Tilde ~
		Double Quotes “

Table 2-1. Character Set

2.1.2 Reserved Words

A Reserved Word has a predetermined meaning to the compiler. The 114 Reserved Words are listed in Table 2-2.

ADDRESS	DC.CODE.LINE	MASTER.CARRIER	SERIAL2.STOP
AND	DC.CUTOUT.TIMER	MASTER.CHECKBACK	SERIAL.BAUD
ARES.FAIL.TIMER	DC.FAIL.TIMER	MASTER.COMMON	SERIAL.BYTE.DLY
ARES.FLAG	DC.FLAG	MASTER.KEYOFF	SERIAL.CARRIER
ARES.GNDLNK	DIAG.CARRIER	MASTER.KEYON	SERIAL.DCD.DELAY
ARES.GROUND	DIAG.FAIL.TIMER	MASTER.PARITY	SERIAL.FLAG
ARES.HEALTH	DIAG.KEYOFF	MASTER.PEER.ADDR	SERIAL.KEYOFF
ARES.KEYOFF	DIAG.KEYON	MASTER.SECURITY	SERIAL.KEYON
ARES.KEYON	DIAG.PASSWORD	MASTER.STOP	SERIAL.PARITY
ARES.LINK	DIAG.PORT.ADDR	MASTER.TIMEOUT	SERIAL.SLAVE
ARES.SLAVE	DIAG.PORT.BAUD	MIN	SERIAL.STOP
ARES.TIME	DT8.PEER.ADDRESS	MSEC	SERIAL.XMT.LIMIT
ASSIGN	DT8.SLV.ADDRESS	NOT	SERIAL2,KEY0FF
ATCS.FAIL.TIMER	END	OR	SERIAL2.BYTE.DLY
ATCS.FLAG	FRAME.LENGTH	OUTPUT	SERIAL2.CARRIER
ATCS.GNDLNK	GENISYS.2000	PEERMODE.BT.ITVL	SET
ATCS.GROUND	IND.ACK.TIMEOUT	PEERMODE.TIMEOUT	SLAVE.PORT.BAUD
ATCS.HEALTH	IND.BRDCST.ITVL	PROGRAM	SPARE
ATCS.LINK	IND.CHANGE.DELAY	QUEUE.OPTION	TIMER
ATCS.MCP	IND.OFFSET	RF.RETRIES	TO
ATCS.MCPLNK	INDICATION	SEC	VALIDATION
ATCS.SLAVE	INPUT	SER.FAIL.TIMER	VAR
BEGIN	INTERFACE	SER2.DCD.DELAY	WIU.ADDRESS
CLEAR	ITEM	SER2.FAIL.TIMER	WORD
CONFIGURATION	LENGTH	SER2.FRME.LENGTH	WORD16
CONTROL	LOCAL	SER2.XMT.LIMIT	WORD32
CONTROL.DELIVERY	LOGIC.VERSION	SERIAL2,KEYON	XOR
CTL.OFFSET	MASTER	SERIAL2.BAUD	
DC.AUTO.RECALL	MASTER.BAUD	SERIAL2.PARITY	

Table 2-2. GENISYS 2000 Reserved Words

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 2-1, and cannot consist of all numbers. A maximum of 16 characters may be used to create symbol names. Examples of legal symbols in G.D.S.2. Versions 1.01 and higher include:

relay.123	DOG
.INPUT RELAY	lTK
IN.TRK.IN	OUT.16

Illegal examples of the above symbols in G.D.S.2. include:

.INPUT.RELAY.4567	(Exceeds 16 characters)
RELAY#1	(Includes an illegal character)
1 2 3 4 5	(all numbers)

2.1.4 Delimiters

Delimiters separate individual words. Every distinct word or token in the source program must be separated by one delimiter. Extra space and tab delimiters may be inserted anywhere in the program they have no effect on the meaning of the program. Delimiters in the non-vital program language are listed in Table 2-3.

space	semicolon (**;**)	open parenthesis (**(**)	tab
equal sign (=)	close parenthesis ())	colon (**:**)	comma (**,**)
carriage return <**CR**>	backslash (\)	percent (**%**)	tilde (~)
at (**@**)	plus sign (+)	Asterisk (*****)	Double Quotes (**”**)

Table 2-3. Delimiters

2.1.5 Formats

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.

2.1.6 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 percent sign (%) and end with a backslash (\). For example:

% THIS IS AN EXAMPLE OF A LEGAL GENISYS 2000 COMMENT

When the compiler encounters the percent sign (%), characters are ignored until it reaches a backslash (\). Switches are the exception (refer to section 2.1.7, Compiler Switches). 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). Another example of a correct comment is as follows:

```
% THIS IS AN EXAMPLE
OF A LEGAL GENISYS
2000 COMMENT\
```

The closing backslash (\) must be inserted, otherwise the compiler ignores all other characters until a backslash is found.

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.

2.1.7 Compiler Switches

Compiler switches are used in the source program to select various options which control the output of the compiler.

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 it encounters the next backslash symbol (\).

Example	Comment
%$D+\	Sets switch “D” to value "ON".
%$D- THIS IS A COMMENT FOLLOWING THE SWITCH\	Sets switch “D” to off. The remainder is a comment.

The compiler switch is as follows: %$Dv\ Debug (where v = + or -)

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

%$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.

2.2 PROGRAM EXAMPLES

2.2.1 Local Input/Output

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

```
GENISYS.2000
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 section declarations in the program source file (PROGRAM, INTERFACE, LOCAL, OUTPUT WORD, OUTPUT WORD32, INPUT WORD, INPUT WORD32) when present, are always in the order shown. These are discussed later in this section.

The INTERFACE section defines all inputs and outputs for the GENISYS 2000 unit. These include both local parallel inputs and outputs, serial inputs and outputs associated with Master and Slave communication port.

The simple example shown above defines only LOCAL inputs and outputs which are processed by 16 bit input and output boards. There is one OUTPUT WORD declaration for each 16-bit output board and one INPUT WORD declaration for each 16 bit input board. (The 32-bit boards require OUTPUT WORD32 and INPUT WORD32 respectively.) A combined total of 16 OUTPUT WORDS and INPUT WORD sections may be declared, the first corresponding to card slot J3 in the GENISYS 2000 cardfile and the 16th corresponding to card slot J18. Output bits are defined within each OUTPUT WORD section (up to 16-bit output board; 32 for a 32-bit output board). The first defined corresponds to bit 0 on the output PCB; the 16th to output bit 15. Inputs are defined in the same manner. If less bits are defined than the input or output boards supports, the remaining bits on that board are treated as spare bits which have no effect on the performance of the application program.

The number, order and position of the I/O boards defined in the program must match the actual hardware configuration. Output boards must be declared first followed by input boards. Positions may be reserved for future output and input boards by declaring SPARE output and input boards. SPARE and unused input bits are ignored; SPARE and unused output bits are automatically assigned a value of 0. These may be 16 or 32 bit boards as required.

The actual system logic is defined with ASSIGN statements. These statements appear between the BEGIN and END declarations. 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 2.4.6, Logic Processing). 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.

2.2.2 Internal Relays and Stick Logic

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

```
GENISYS.2000
PROGRAM NBR2;
INTERFACE
LOCAL
OUTPUT WORD:
        OUT.1, OUT.2, OUT.3, OUTA, 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 the Local I/O section, however OUT.6 is added as the 6th output on the 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 de-energized (logical “0”). 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 de-energized. The internal STICK and OUT.6 will remain energized until IN.D is energized. This results in the clearing of the stick circuit. However, that if IN.C is still energized, STICK and OUT.6 will remain energized.

2.2.3 Timing Relays

The sample program below shows the handling of timing relays. This program makes use of one timer relay. Continuing from the Program Example section, two more defined output bits, OUT.7 and OUT.8 are added. In the VAR section, another internal relay (Tl) 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.

```
GENISYS.2000
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 = l: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 OUTA;
         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. 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.

2.2.4 Master/Slave Communications

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

```
GENISYS.2000
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.OUTA, 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:

```
GENISYS.2000
PROGRAM NBR45;
INTERFACE
        LOCAL
                OUTPUT WORD:
                        OUT.1, OUT.2, OUT.3; SERIAL SLAVE
        SERIAL.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 the section 2.3.2, Interface.

2.3 DETAILED STATEMENT DESCRIPTIONS

2.3.1 Program Statement

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

FORMAT:

GENISYS.2000 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 2.1.3, User-Defined Symbols). Comments may be placed before a program statement.

2.3.2 Interface

The INTERFACE section contains the various output and input specifications of the system. There are five possible subsections in the INTERFACE section: LOCAL I/O, MASTER I/O port, SERIAL.SLAVE I/O port, SERIAL.SLAVE2 I/O port, ATCS.SLAVE I/O port, ARES SLAVE I/O port and the DC.CODE.LINE I/O port. These correspond to the five output/input interfaces on the controller board. At least one of the five I/O interfaces must be defined. Each begins with a name (LOCAL, MASTER and any of the various Slave types: SERIAL.SLAVE, SERIAL.SLAVE2, ATCS.SLAVE, ARES.SLAVE, DC.CODE.LINE) that designates the type of interface. The LOCAL I/O is defined first, followed by MASTER and then the various Slave types. Select only one Slave type per application. Within each of these, outputs are defined first, followed by inputs if any. Either outputs or inputs may be blank, but not both at the same time.

2.3.3 Local Section

The LOCAL 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 2000 - 1 to 16 words may be defined, with 1 to 16 or 1 to 32 bit names on each word
(board type dependent).

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 identify 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

All input boards are scanned every 50 milliseconds. Data delivery to output boards occurs as required. Pulse duration for 16-bit pulse delivery boards is selectable in the configuration section of the application program.

```
GENISYS.2000 PROGRAM id;
INTERFACE
LOCAL
OUTPUT
WORD : <id list>;
```

GENISYS 2000: Up to 16 boards. <id list>
(You may have up to 16 or 32 relay names, dependent on board type).

2.3.4 MASTER

The MASTER section defines the input/output interface for each remote GENISYS protocol Slave with which the GENISYS 2000 MASTER serial port communicates. The definition of each Slave begins with an ADDRESS declaration which identifies the remote Slave address. Following the address declaration is the OUTPUT section which defines bits passed from the Master GENISYS 2000 unit to the GENISYS protocol. The INPUT section defines bits passed from the remote GENISYS protocol Slave to the Master unit. The MASTER section may contain up to 64 ADDRESS declarations, each one defining the interface with a different GENISYS 2000 Slave. An example follows on the next page:

```
INTERFACE
    .
    .
    .
    MASTER

          ADDRESS:1

                  OUTPUT:

                        OUT1.1, OUT1.2, OUT1.3, OUT1.4,...OUT1.n;

                  INPUT:

                        IN1.1, IN1.2, OUT1.3, OUT1.4, ... OUT1.n;

          ADDRESS:        3

                  OUTPUT:

                        OUT3.1, OUT3.2, OUT3.3, OUT3.4,...OUT3.n;

                  INPUT:

                        IN3.1, IN3.2, IN3.3 ...IN3.n;

         .
         .
         .
    SERIAL.SLAVE
    .
    .
    .
```

2.3.4.1 ADDRESS Declaration

The ADDRESS declaration within the MASTER section defines a remote GENISYS protocol Slave that communication is to be implemented and specifies its address. Valid addresses are 1 to 255 decimals and the addresses do not have to be consecutive. Definition of bits which are sent to the Slave and received from the Slave follow the address declaration. All defined Slaves must have either an OUTPUT or INPUT declaration as a minimum. In the normal case, both declarations are present.

2.3.4.2 OUTPUT Declaration

OUTPUT declaration immediately follows the ADDRESS declaration if outputs are to be defined. The OUTPUT declaration is not required. Following the OUTPUT declaration is a list of bits which are to be sent to the Slave. The list may contain 1 to 256 bit names separated by commas and terminated by a semicolon.

2.3.4.3 INPUT Declaration

INPUT declaration immediately follows the OUTPUT declaration or the ADDRESS declaration if the OUTPUT declaration is not present. The INPUT declaration is not required. Following the INPUT declaration is a list of bits which are expected to be received from the Slave unit. The list may contain 1 to 256 bit names separated by commas and terminated by a semicolon.

2.3.5 SERIAL.SLAVE

The SERIAL.SLAVE section defines the input/output interface for the GENISYS 2000 Serial Slave communication port. This declaration is valid only in application programs to be used with GENISYS 2000 Executives that support a Serial Slave communication port. (Presently, these include, the GENISYS, MCS-1, DATATRAIN II and VIII, DF1 and WB&S S2 Executives.) These various code types are declared immediately after the keyword SERIAL.SLAVE:. The code type is entered as a “string” which is preceded and followed with a double quote . The valid code types are “GENISYS”, “DF1”, “GRSDTS”, “GRSDTP”, “GRSDT2”, “GRSDT2C”, “MCS1”, “MCS2” and “WBSS2”. This declaration is optional. The default is “GENISYS”. A GENISYS 2000 unit may be programmed to simultaneously represent up to 6 different Serial Slaves. This capability is normally used when emulating code formats with limited control and indication message lengths (MCS-1, for example) to expand the Slave communication capacity of a single GENISYS 2000 unit. While multiple Slave support is not required for the GENISYS Slave protocol, it is still available. Additionally, an independent dualport configuration may be defined using the keyword SERIAL.SLAVE2. All logic for this type is identical to the SERIAL.SLAVE type. Refer to the SERIAL2 and SER2 configuration items for port specifics. Note, SERIAL.SLAVE2 refers specifically to port 2.

The definition of each “logical Slave” begins with an ADDRESS declaration which identifies the Slave address. Following the ADDRESS declaration is the OUTPUT section which defines bits passed from the logical Slaves to its Master. The INPUT section follows the OUTPUT section and defines bits passed from the Master to the logical Slave. The SERIAL.SLAVE (and or SERIAL.SLAVE2) section may contain up to 6 ADDRESS declarations, each one defining a different Serial Slave. An example is shown on the next page:

```
INTERFACE
        .
        .
        .
      SERIAL.SLAVE: “GENISYS”      % SERIAL CODE TYPE IS AN OPTIONAL PARAMETER\

        ADDRESS: 1

                OUTPUT:

                        SLVOUT1.1, SLVOUT1.2, SLVOUT1.3, SLVOUT1.4, …SLVOUT1.n;

                INPUT:
                        SLVIN1.1, SLVIN1.2, SLVOUT1.3, SLVOUT1.4, … SLVOUT1.n;

        ADDRESS:  3

                OUTPUT:

                        SLVOUT3.1, SVLOUT3.2, SVLOUT3.3, SVLOUT3.4, …SVLOUT3.n;
                INPUT:

                        SLVIN3.1, SLVIN3.2, SLVIN3.3 … SLVIN3.n;

                .
                .
                .
        CONFIGURATION
```

2.3.5.1 ADDRESS Declaration

The ADDRESS declaration within the SERIAL.SLAVE (and or SERIAL.SLAVE2) section defines a “logical Slave” and specifies its address. Valid addresses vary with the Serial Slave protocol implemented. For GENISYS and WB&S S2 protocol Slaves, valid addresses are 1 to 255 decimal. Valid addresses for MCS-1 protocol Slaves are 1 to 62 decimal. The Slave addresses do not have to be consecutive. Definition of bits which are sent to the Master and received from the Master follow the address declaration. All defined logical Slaves must have either an OUTPUT or INPUT declaration as a minimum. In the normal case both declarations are present.

2.3.5.2 FLAG Declaration

The SERIAL.FLAG declaration is OPTIONAL. This parameter is defaulted to zero. It will be used in future development.

2.3.5.2.1 DATATRAIN VIII Specific

An optional parameter for the DATATRAIN VIII code system follows: The DATATRAIN VIII serial code type may require a specific one to four byte hexadecimal address which is specified after the optional **SERIAL.FLAG** parameter and before the **OUTPUT** keyword. Either a slave address and or a peer address may be declared. Defaults for slave and peer addresses are zero. If this parameter is specified the slave address is specified by using the keyword **DT8.SLV.ADDRESS:** followed by a variable length (1 to 4 byte) hexadecimal string which is preceded and followed by double quotes. It is also possible to specify the peer address by using the keyword **DT8.PEER.ADDRESS:** followed by a variable length (1 to 4 byte) hexadecimal string which is preceded and followed by double quotes.

2.3.5.2.2 ALLEN BRADLEY DF1 Specific

An optional parameter for the DF1 code system follows: The DF1 serial code type may require a specific control and indication byte offset which is specified after the optional **SERIAL.FLAG** parameter and before the **OUTPUT** keyword. Either a control offset and or a indication offset may be declared. Defaults for control and indication offsets are zero. If this parameter is specified the control offset is specified by using the keyword **CONTROL.OFFSET:** followed by an integer value in the range of 0-99 bytes. It is also possible to specify the indication offset by using the keyword **IND.OFFSET:** followed by an integer value in the range of 0-99 bytes.

2.3.5.3 OUTPUT Declaration

The OUTPUT declaration immediately follows the ADDRESS declaration if outputs are to be defined. The OUTPUT declaration is not required. Following the OUTPUT declaration is a list of bits which are to be sent to the Master. The list may contain 1 to 256 bit names separated by commas and terminated by a semicolon. (The upper limit of 256 bits applies to GENISYS protocol Slaves. The limit for other Serial Slave types may be lower.)

2.3.5.4 INPUT Declaration

INPUT declaration immediately follows the OUTPUT declaration or the ADDRESS declaration if the OUTPUT declaration is not present. Following the INPUT declaration is a list of bits which are expected to be received by the Slave. The list may contain 1 to 256 bit names separated by commas and terminated by a semicolon. (The upper limit of 256 bits applies to GENISYS protocol Slaves. The limit for other Serial Slave types may be lower.)

2.3.6 ATCS.SLAVE or ARES.SLAVE

The ATCS.SLAVE or ARES.SLAVE section defines the input/output interface for the GENISYS 2000 AxxS Slave communication port. This declaration is valid only in application programs to be used with GENISYS 2000 AxxS Slave protocol Executive. Up to 6 AxxS "logical Slaves" can be supported by a single GENISYS 2000 unit although only one is normally defined. The "xx" refers to ATCS or ARES Slave types; ATCS and ARES may not be mixed. In future releases, various ATCS/ARES slave types may be declared immediately following the AxxS.SLAVE keyword. Presently, the ARES slave type is "ARES" and the ATCS slave type is "ATCS". This optional parameter will follow the same syntax as SERIAL.SLAVE and DC.SLAVE. Any mixing of these two Slave types will result in a compiler error. (See section 2.3.5, SERIAL.SLAVE for additional discussion on the definition of multiple logical Slaves.)

The definition of each “logical Slave” begins with an ADDRESS declaration. Following the ADDRESS declaration is the WIU.ADDRESS declaration. Following the WIU.ADDRESS is the OUTPUT section which defined bits passed from the logical Slave to its Master. The INPUT section follows the OUTPUT section and defines bits passed from the Master to the logical Slave. The ATCS/ARES.SLAVE section may contain up to 6 ADDRESS declarations, each one defining a different ATCS/ARES.Slave. An example is shown here:

```
INTERFACE
        .
        .
        ATCS.SLAVE
                ATCS.FLAG:0;

                ADDRESS
                        WIU.ADDRESS:            “5266A1A1A100000A”

                OUTPUT:

                        SLVOUT1.1, SLVOUT1.2, SLVOUT1.3, SLVOUT1.4, …SLVOUT1.n;

                INPUT:

                        SLVIN1.1, SLVIN1.2, SLVOUT1.3, SLVOUT1.4, … SLVOUT1.n;

                ADDRESS
                        WIU.ADDRESS:            ”5266A1A1A300000A”

                OUTPUT:

                        SLVOUT3.1, SVLOUT3.2, SVLOUT3.3, SVLOUT3.4, …SVLOUT3.n;

                INPUT:

                        SLVIN3.1, SLVIN3.2, SLVIN3.3 … SLVIN3.n;
 CONFIGURATION
 .
 .
```

2.3.6.1 FLAG Declaration

The ATCS.FLAG or ARES.FLAG declaration is OPTIONAL. This parameter is defaulted to zero. It will be used in future development.

2.3.6.2 ADDRESS Declaration

The ADDRESS declaration within the AxxS.SLAVE section defines a “logical Slave”. The ADDRESS declaration requires no parameters in the AxxS Slave section. The WIU.ADDRESS declaration follows the ADDRESS declaration and specifies the AxxS address of the logical Slave. Definition of bits which are sent to the Master and received from the Master follow the WIU.ADDRESS declaration. All defined logical Slaves must have either an OUTPUT or INPUT declaration as a minimum. In the normal case, both declarations are present.

2.3.6.3 WIU.ADDRESS Declaration

The WIU.ADDRESS declaration specifies the ATCS address for the logical Slave. This address is a 16 digit number enclosed in double quotes (“). The format of this address is as specified in the latest revision of ATCS (PTS) Specification 200. In general, the format of this address is "IRRRWWWWWW00000L" where:

I	address type identifier from ATCS (PTS) Spec 200 , appendix T
RRR	railroad ID from ATCS (PTS) Spec. 250, Appendix F
WWWWWW	field code unit identity (the format of this segment is determined by the architecture of the specific communication network being used)
L	a hexadecimal character representing the number of valid digits in the address (A to F)

Per the ATCS address syntax rules, the address digits are left justified, "0" digits are replaced by “A”, and unused digits are “0”. The maximum legal length of an ATCS address is 15 digits. The 16th digit is always the number of used (non-zero) digits in the address.

Per the ARES address syntax rules, the address digits are left justified and zero (0) digits are valid in the address. The maximum legal length of the ARES address is 15 digits. The 16th digit is always the number of digits in the address.

2.3.6.4 OUTPUT Declaration

The OUTPUT declaration immediately follows the WIU.ADDRESS declaration if outputs are to be defined. Following the OUTPUT declaration is a list of bits which are to be sent to the Master. The list may contain 1 to 256 bit names separated by commas and terminated by a semicolon.

2.3.6.5 INPUT Declaration

INPUT declaration immediately follows the OUTPUT declaration or the ADDRESS declaration if the OUTPUT declaration is not present. The INPUT declaration is not required. Following the INPUT declaration is a list of bits which are expected to be received by the Slave. The list may contain 1 to 256 bit names separated by commas and terminated by a semicolon.

2.3.7 DC.CODE.LINE

The DC.CODE.LINE section defines the input/output interface for the GENISYS 2000 DC code line Slave communication port. This declaration is valid only in application programs to be used with GENISYS 2000 DC code line Slave Executive. Following the DC.CODE.LINE keyword and a colon is a code line type enclosed in double quotes (“). Currently, the valid types are USS504, USS506, and USS514. Up to 6 DC code line “logical Slaves” (PUPS) can be supported by a single GENISYS 2000 unit. (See SERIAL.SLAVE section for additional discussion on the definition of multiple logical Slaves.)

The definition of each “logical Slave” begins with an ADDRESS declaration. Following the ADDRESS declaration is the INDICATION declaration. This declaration defines the indication (field to office) section of the DC Slave. The indication message LENGTH, indication ADDRESS, OUTPUT declaration follow in sequence.

The CONTROL declaration defines the control (office to field) section of the DC Slave. The control message LENGTH, control ADDRESS and INPUT declaration follow in sequence. The DC.CODE.LINE section may contain up to 6 ADDRESS declarations, each one defining a different DC code line Slave. An example is shown on the next page:

```
INTERFACE
         .
         .
         .
         DC.CODE.LINE:                "USS514"

                  ADDRESS

                           INDICATION
                           LENGTH:                16;
                           ADDRESS:               "SSLSLSLS"

                           OUTPUT:

                                    SLVOUT1.1, SLVOUT1.2, SLVOUT1.3, SLVOUT1.4, …SLVOUT1.n;
                           CONTROL
                           LENGTH:                16;
                           ADDRESS:               "SSLSLSLS"

                           INPUT:

                                    SLVIN1.1, SLVIN1.2, SLVOUT1.3, SLVOUT1.4, … SLVOUT1.n;

                  ADDRESS
                           INDICATION
                           LENGTH:                16;
                           ADDRESS:               "SSLSLLSS"

                           OUTPUT:

                                    SLVOUT3.1, SVLOUT3.2, SVLOUT3.3, SVLOUT3.4, …SVLOUT3.n;
                           CONTROL
                           LENGTH:                16
                           ADDRESS:               "SSLSLLSS"




                           INPUT:

                                    SLVIN3.1, SLVIN3.2, SLVIN3.3 … SLVIN3.n;

                  .
                  .
                  .
         CONFIGURATION
         .
         .
```

2.3.7.1 FLAG Declaration

The DC.FLAG declaration is OPTIONAL. This parameter is defaulted to zero. It will be used in future development.

2.3.7.2 ADDRESS Declaration

The ADDRESS declaration within the DC.CODE.LINE section defines a “logical Slave”. The ADDRESS declaration requires no parameter in the DC Slave section.

2.3.7.3 INDICATION Declaration

The INDICATION declaration marks the beginning of the section in which the indication characteristics of the DC Slave are specified. This section includes the indication message LENGTH, the indication ADDRESS, and the OUTPUT list which specifies the bits contained in the indication message for this Slave. The INDICATION declaration includes no parameters.

2.3.7.4 Indication LENGTH Declaration

The indication LENGTH declaration specifies the length of the indication message. The length of the indication message includes the address (always 8 bits), the number of indication bits to be transmitted, and 1 terminator bit. If the resulting value is an odd number, one additional bit is added. Valid values are even numbers between 10 and 64 for USS514 code lines and 16 for all other supported code lines. If the indication LENGTH declaration is omitted, the indication length defaults to 16. Notice that the specified length need not match the number of bits in the OUTPUT list, but it should be greater than the number of bits in the OUTPUT list. If the length specified is insufficient to transport all indication bits defined for the DC Slave, bits at the end of the OUTPUT list will not be included in the indication message. Bits transmitted in the indication message beyond those defined in the OUTPUT list (the indication message includes more bits than are defined) are always transmitted as short (“0”) code steps. In any case, the specified indication length must match that specified for the office code unit or misoperation will result.

2.3.7.5 Indication ADDRESS Declaration

The indication ADDRESS declaration specifies the address to be transmitted with the indication for this Slave. The ADDRESS is always 8 bits (steps) long and must be a valid 504, 506, 514 address. The address is specified in Long/Short (L/S) or 1/0 format and must be enclosed in double quotes (“). See the appropriate code system manual for valid addresses.

2.3.7.6 OUTPUT Declaration

The OUTPUT declaration immediately follows the indication ADDRESS declaration if outputs are to be defined. Following the OUTPUT declaration is a list of bits which are to be sent to the office code unit. The list may contain 1 to 49 bit names for USS514 Slave, 1 to 7 bit names for a USS504 or USS506 Slave. The bit names are separated by commas. The list is terminated by a semicolon.

2.3.7.7 CONTROL Declaration

The CONTROL declaration marks the beginning of the section in which the control characteristics of the DC Slave are specified. This section includes the control message LENGTH, the control ADDRESS (normally the same as the indication ADDRESS), and the INPUT list which specifies the bits contained in the control message for this Slave. The CONTROL declaration includes no parameters.

2.3.7.8 Control LENGTH Declaration

The control LENGTH declaration specifies the expected length of the control message. The length of the control message includes the address (always 8 bits), the number of control bits expected in a control message, and 1 terminator bit. If the resulting value is an odd number, one additional bit is added. Valid values are even numbers between 10 and 64 for USS514 code lines and 16 for all other code lines. If the control LENGTH declaration is omitted, the control length defaults to 16. Note that the specified length need not match the number of bits in the INPUT list, but it must be greater than the number of bits in the INPUT list. Bits received in the control message beyond those in the INPUT list are discarded. If less than the specified number of bits are received, the entire message is discarded. In any case, the specified control length must match that specified for the office code unit or misoperation will result.

2.3.7.9 Control ADDRESS Declaration

The control ADDRESS declaration specifies the address associated with the input bits for this Slave. The address is always 8 bits (steps) long and must be a valid 504, 506, 514 address. It should (but is not required to) match the indication ADDRESS. The address is specified in Long/Short (L/S) or 1/0 format and must be enclosed in double quotes(“). See the appropriate code system manual for valid addresses.

2.3.7.10 INPUT Declaration

The INPUT declaration immediately follows the control ADDRESS declaration. Following the INPUT declaration is a list of bits which are expected to be received by the Slave. The list may contain 1 to 49 bit names for a USS 514 Slave or 1 to 7 bit name for a USS 504 or USS 506 Slave. The bit names are separated by commas. The list is terminated by a semicolon.

2.4 CONFIGURATION ITEMS FOR PROGRAM PARAMETERS

ARES.FAIL.TIMER

APPLICABILITY:	ARES.SLAVE
RANGE OF VALUES:	10 TO 600 SECONDS
DEFAULT:	300 SECONDS
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

ARES.FAIL.TIMER specifies the time in seconds after which the ARES communication link is declared failed when no message addressed to any of the defined ARES Slaves have been detected. When the ARES fail timer has expired, the system bit ARES.MASTER.ON is cleared. This bit can be used by the application program to obtain current status of the ARES communication link.

ARES.GNDLNK

APPLICABILITY:	ARES.SLAVE
RANGE OF VALUES:	VALUES BETWEEN 01 AND FE (HEX)
DEFAULT:	2
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

ARES.GNDLNK sets the HDLC link address to which the ARES Slave port addresses its responses. The default should not be changed.

ARES.GROUND

APPLICABILITY:	ARES.SLAVE
RANGE OF VALUES:	SEE “SPECIFICATION OF ARES ADDRESSES”
DEFAULT:	0 (NULL ADDRESS)
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

ARES.GROUND sets the ARES ground host address to which Slave data is sent and from which Slave data is accepted. This address must be specified if ARES Slaves are defined. If not specified in the application program, the ARES ground host address must be entered during the local configuration procedure which is part of the installation process.

ARES.HEALTH

APPLICABILITY:	ARES.SLAVE
RANGE OF VALUES:	SEE “SPECIFICATION OF ARES ADDRESSES”
DEFAULT:	0 (NULL ADDRESS)
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

ARES.HEALTH sets the ARES address to which unit status (health) information is sent. If it is not specified, no status information is sent.

ARES.KEYON

APPLICABILITY:	ARES.SLAVE
RANGE OF VALUES:	0 TO 300 BIT TIMES
DEFAULT:	300
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

ARES.KEYON sets the delay between carrier key-on (request to send) and the beginning of transmitted data. The required key-on delay value is dictated by the characteristics of the communication circuit to which the Slave port is connected.

ARES.KEYOFF

APPLICABILITY:	ARES.SLAVE
RANGE OF VALUES:	0 TO 280 BIT TIMES
DEFAULT:	280
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

ARES.KEYOFF sets the delay between the end of transmitted data and the deassertion of request to send. The required key-off delay value is dictated by the characteristics of the communication circuit to which the Slave port is connected.

ARES.LINK

APPLICABILITY:	ARES.SLAVE
RANGE OF VALUES:	ODD VALUES BETWEEN 01 AND FF(HEX)
DEFAULT:	1
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

ARES.LINK sets the HDLC link address to which the ARES Slave port responds. The default is the value assigned and should not be changed.

ARES.TIME

APPLICABILITY:	ARES.SLAVE
RANGE OF VALUES:	SEE “SPECIFICATION OF ARES ADDRESSES”
DEFAULT:	0 (NULL ADDRESS)
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

ARES.TIME sets the ARES address from which unit status (time) information is requested. If it is not specified, no standard timing information is requested.

ATCS.FAIL.TIMER

APPLICABILITY:	ATCS.SLAVE
RANGE OF VALUES:	10 TO 600 SECONDS
DEFAULT:	300 SECONDS
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

ATCS.FAIL.TIMER specifies the time in seconds after which the ATCS communication link is declared failed when no message addressed to any of the defined ATCS Slaves have been detected. When the ATCS fail timer has expired, the system bit ATCS.MASTER.ON is cleared. This bit can be used by the application program to obtain current status of the ATCS communication link.

ATCS.GNDLNK

APPLICABILITY:	ATCS.SLAVE
RANGE OF VALUES:	ODD VALUES BETWEEN 01 AND FE (HEX)
DEFAULT:	23
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

ATCS.GNDLNK sets the HDLC link address to which the ATCS Slave port addresses its responses. The default is the value assigned by ATCS Spec. 200 Appendix K. It should not be changed.

ATCS.GROUND

APPLICABILITY:	ATCS.SLAVE
RANGE OF VALUES:	SEE “SPECIFICATION OF ATCS ADDRESSES”
DEFAULT:	0 (NULL ADDRESS)
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

ATCS.GROUND sets the ATCS ground host address to which Slave data is sent and from which Slave data is accepted. This address must be specified if ATCS Slaves are defined or the GENISYS 2000 unit will not establish communications with the MCP. If not specified in the application program, the ATCS ground host address must be entered during the local configuration procedure which is part of the installation process.

ATCS.HEALTH

APPLICABILITY:	ATCS.SLAVE
RANGE OF VALUES:	SEE “SPECIFICATION OF ATCS ADDRESSES”
DEFAULT:	0 (NULL ADDRESS)
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

ATCS.HEALTH sets the ATCS address to which unit status (health) information is sent. f it is not specified, no status information is sent.

ATCS.LINK

APPLICABILITY:	ATCS.SLAVE
RANGE OF VALUES:	ODD VALUES BETWEEN 01 AND FF(HEX)
DEFAULT:	3
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

ATCS.LINK sets the HDLC link address to which the ATCS Slave port responds. The default is the value assigned by ATCS Spec. 200, Appendix K. It should not be changed.

ATCS.MCP

APPLICABILITY:	ATCS.SLAVE
RANGE OF VALUES:	SEE “SPECIFICATION OF ATCS ADDRESSES”
DEFAULT:	0 (NULL ADDRESS)
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

ATCS.MCP sets the ATCS address to be used by the MCP. This address must be specified if ATCS Slaves are defined or the connected MCP will not complete its initialization and become operational. If not specified in the application program, the MPC ATCS address must be entered during the local configuration procedure which is part of the installation process.

ATCS.MCPLINK

APPLICABILITY:	ATCS.SLAVE
RANGE OF VALUES:	ODD VALUES BETWEEN 01 AND FF(HEX)
DEFAULT:	1
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

ATCS.MCPLINK sets the HDLC link address to which the ATCS Slaveport addresses the MCP for supervisory purposes. The default is the value assigned by ATCS Spec. 200, Appendix K. It should not be changed.

CONTROL.DELIVERY

APPLICABILITY:	ALL GENISYS 2000 APPLICATIONS
RANGE OF VALUES:	10 TO 4000 milliseconds
DEFAULT:	50
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

This value is the duration of the controller PCB delivery pulse to the relay PCB’s. This parameter controls the output on time for pulse delivery boards. It has no effect on continuous delivery boards. Continuous delivery boards should be set at the default setting.

DC.AUTO.RECALL

APPLICABILITY:	DC.SLAVE
RANGE OF VALUES:	ENABLED,DISABLED
DEFAULT:	ENABLED
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

DC.AUTO.RECALL determines whether or not a DC Slave will respond to a valid received control with an indication message. When DC.AUTO.RECALL is enabled, the DC Slave responds with an indication whenever it received a valid control. When it is disabled no response is generated.

DC.CUTOUT.TIMER

APPLICABILITY:	DC.SLAVE
RANGE OF VALUES:	0 (disabled) to 300 seconds
DEFAULT:	180 seconds
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

DC.CUTOUT.TIMER sets the length of time that the DC protocol port may transmit indications continuously. If this time limit is exceeded, indication transmission ceases for a time period equal to the cutout timer or until an indication recall addressed to one of the defined DC Slaves is received. If the timer value is set to 0 this function is disabled.

DC.FAIL.TIMER

APPLICABILITY:	DC.SLAVE
RANGE OF VALUES:	10 to 600 seconds
DEFAULT:	300 seconds
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

DC.FAIL.TIMER specifies the time in seconds after which the DC communication link will be declared failed if no valid messages are received in the DC code line port. When the DC fail timer expires, the system bit DC.MASTER.ON is cleared. The application program can use this bit to determine the current status of the DC communication link.

DIAG.CARRIER

APPLICABILITY:	ALL GENISYS 2000 APPLICATIONS
RANGE OF VALUES:	”KEYED or “CONTINUOUS”
DEFAULT:	”CONTINUOUS”
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

DIAG.CARRIER sets the type of communication channel to be used by the diagnostic serial port. “CONTINUOUS” carrier is used on a communication channel with full-duplex capability while “KEYED” carrier is used on a half-duplex communication channel. “KEYED” and “CONTINUOUS” refer to the state of the carrier signal leaving the Master (office) end of the communication circuit.

DIAG.FAIL.TIMER

APPLICABILITY:	ALL GENISYS 2000 APPLICATIONS
RANGE OF VALUES:	10 TO 600 SECONDS
DEFAULT:	300 SECONDS
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

DIAG.FAIL.TIMER specifies the time in seconds after which the serial communication link is declared failed when no messages addressed to the specified GENISYS 2000 diagnostic port address have been detected.

DIAG.KEYOFF

APPLICABILITY:	ALL GENISYS 2000 APPLICATIONS
RANGE OF VALUES:	0 TO 300 BIT TIMES
DEFAULT:	0
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

DIAG.KEYOFF sets the delay between the end of transmitted data and the deassertion of request to send. The required key-off delay value is dictated by the characteristics of the communication circuit to which the diagnostic port is connected.

DIAG.KEYON

APPLICABILITY:	ALL GENISYS 2000 APPLICATIONS
RANGE OF VALUES:	0 TO 300 BIT TIMES
DEFAULT:	0
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

DIAG.KEYON sets the delay between carrier key-on (request to send) and the beginning of transmitted data. The required key-on delay value is dictated by the characteristics of the communication circuit to which the diagnostic port is connected.

DIAG.PASSWORD

APPLICABILITY:	ALL GENISYS 2000 APPLICATIONS
RANGE OF VALUES:	ANY ALPHANUMERIC COMBINATION (1 TO 8 LENGTH)
DEFAULT:	“PASSWORD”
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

DIAG.PASSWORD enables the application to define specific passwords. The password may be composed of any combination of (A through Z) and (0 through 9) and space. The minumum length is one and the maximum length is eight.

DIAG.PORT.ADDR

APPLICABILITY:	ALL GENISYS 2000 APPLICATIONS
RANGE OF VALUES:	1-255
DEFAULT:	1
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

DIAG.PORT ADDR sets the specific address associated with the diagnostic port. This address is used for remote diagnostic access to the GENISYS 2000 unit.

DIAG.PORT.BAUD

APPLICABILITY:	ALL GENISYS 2000 APPLICATIONS
RANGE OF VALUES:	300, 600, 1200, 2400, 4800, 9600 BITS PER SECOND
DEFAULT:	2400 BITS PER SECOND
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

DIAG.PORT.BAUD sets the data rate to be used by the GENISYS 2000 diagnostic port.

FRAME.LENGTH

APPLICABILITY:	SERIAL.SLAVE (WB&S S2 protocol only)
RANGE OF VALUES:	32, 49, 64, 128 bits
DEFAULT:	32
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

FRAME.LENGTH sets the number of data bits in the WB&S S2 protocol control and indication frames to be processed by the defined WB&S S2 Slaves. This value matches that set in the WB&S S2 office code unit for the code line.

IND.ACK.TIMEOUT

APPLICABILITY:	ARES.SLAVE
RANGE OF VALUES:	5 TO 180 SECONDS
DEFAULT:	10 SECONDS
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

IND.ACK.TIMEOUT sets the timer that an ARES slave will wait for an indication message to be acknowledged by a service signal.

IND.ACK.TIMEOUT

APPLICABILITY:	ATCS.SLAVE
RANGE OF VALUES:	5 TO 180 SECONDS
DEFAULT:	30 SECONDS
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

IND.ACK.TIMEOUT sets the timer that an ATCS slave will wait for an indication message to be acknowledged by a service signal. Care should be taken to ensure that this time is greater than the time required for the MCP to exhaust its transmission retries.

IND.BRDCST.ITVL

APPLICABILITY:	ARES.SLAVE
RANGE OF VALUES:	0, 30 TO 600 SECONDS
DEFAULT:	1500 SECONDS
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

IND.BRDCST.ITVL sets the time between indication broadcasts from an ARES slave. A setting of "0" disables indication broadcasts.

IND.BRDCST.ITVL

APPLICABILITY:	ATCS.SLAVE
RANGE OF VALUES:	0, 30 TO 600 SECONDS
DEFAULT:	60 SECONDS
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

IND.BRDCST.ITVL sets the time between indication broadcasts from an ATCS slave. A setting of "0" disables indication broadcasts.

IND.CHANGE.DELAY

APPLICABILITY:	ALL GENISYS 2000 APPLICATIONS WHICH DEFINE LOCAL OUTPUTS
RANGE OF VALUES:	50 TO 2000 milliseconds
DEFAULT:	1000
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

IND.CHANGE.DELAY sets the delay between the detection of a local input change and the time it is processed by the GENISYS 2000 Executive. This delay is useful in filtering out unwanted input transients.

LOGIC.VERSION

APPLICABILITY:	ALL GENISYS 2000 APPLICATIONS
RANGE OF VALUES:	0 TO 32000
DEFAULT:	0
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

LOGIC.VERSION is set at the application programmer's discretion to identify the current version of the application program. This value is loaded into the application program prom set from which it can be retrieved with the GENISYS 2000 diagnostic tool for identification purposes. It has no other function.

MASTER.BAUD

APPLICABILITY:	MASTER
RANGE OF VALUES:	300, 600, 1200, 2400, 4800, 9600, 19200, BITS PER SECOND
DEFAULT:	300 BITS PER SECOND
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

MASTER. BAUD sets the data rate to be used by the Master serial port.

MASTER.CARRIER

APPLICABILITY:	MASTER
RANGE OF VALUES:	“KEYED” or “CONTINUOUS”
DEFAULT:	“CONTINUOUS”
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

MASTER.CARRIER sets the type of communication channel to be used by the Master serial port. “CONTINUOUS” carrier is used on a communication channel with full-duplex capability while “KEYED” carrier is used on a half-duplex communication channel. “KEYED” and “CONTINUOUS” refer to the state of the carrier signal leaving the Master port.

MASTER.CHECKBACK

APPLICABILITY:	MASTER
RANGE OF VALUES:	“ENABLED” or “DISABLED”
DEFAULT:	“DISABLED”
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

MASTER.CHECKBACK selects the method of control delivery to be used by the Master serial port. The use of checkback control delivery causes data messages from the Master port to field units to be confirmed before they are executed. This slightly improves the security of data messages passing from the Master port to field stations.

MASTER.COMMON

APPLICABILITY:	MASTER
RANGE OF VALUES:	“ENABLED” or “DISABLED”
DEFAULT:	“DISABLED”
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

MASTER.COMMON enables or disables the use of common controls mode on the Master port. Common controls are control messages which are transmitted to all locations at the same time. This feature is provided for compatibility with older GENISYS field units. Its use is not recommended.

MASTER.KEYOFF

APPLICABILITY:	MASTER
RANGE OF VALUES:	0 TO 63 BIT TIMES
DEFAULT:	0
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

MASTER.KEYOFF sets the delay between the end of transmitted data and carrier key-off (removal of request to send). Since the Master port is normally implemented using continuous carrier, Master port key-off delay are not usually necessary. If the Master port is connected to a half-duplex circuit, the Master port key-off delay must be set to a non-zero value and MASTER.CARRIER set to “KEYED”.

MASTER.KEYON

APPLICABILITY:	MASTER
RANGE OF VALUES:	0 TO 63 BIT TIMES
DEFAULT:	0
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

MASTER.KEYON sets the delay between carrier key-on (request to send) and the beginning of transmitted data. Since the Master port is normally implemented using continuous carrier, Master port key-on delay is not usually necessary. If the Master port is connected to a half-duplex circuit, the Master port key-on delay must be set to a non-zero value and MASTER.CARRIER set to “KEYED”.

MASTER.PARITY

APPLICABILITY:	MASTER
RANGE OF VALUES:	“ODD”, “EVEN”, “NONE”
DEFAULT:	NONE
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

MASTER.PARITY sets, transmits and receives parity for the Master serial port. This parameter should be left at its default setting.

MASTER.PEER.ADDR

APPLICABILITY:	DF1
RANGE OF VALUES:	0-255
DEFAULT:	5
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

MASTER.PEER.ADDR sets the address used by the DF1 Slave protocol executive used in indication messages passed from the field to the office.

MASTER.SECURITY

APPLICABILITY:	MASTER
RANGE OF VALUES:	“ON” or “OFF”
DEFAULT:	”OFF”
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

MASTER.KEYOFF selects the type of polling message to be used by the Master serial port. If “ON” is selected, the polling message contains checksum. If “OFF” is selected, the polling message contains no checksum. Communication through the Master port is slightly faster with security “OFF”. With security “ON”, indication data passing from the field units to the Master port is slightly more secure. MASTER.SECURITY should be set “ON” when the security of indication data is an important consideration.

MASTER.STOP

APPLICABILITY:	MASTER
RANGE OF VALUES:	1 or 2
DEFAULT:	1
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

MASTER.STOP sets the number of stop bits to be transmitted by the Master port, “1” is normally sufficient. Transmission of two stop bits may be necessary to accommodate certain timing anomalies created by communication equipment. The Master serial port receiver always requires only 1 stop bit regardless of the stop bit setting.

MASTER.TIMEOUT

APPLICABILITY:	MASTER
RANGE OF VALUES:	30 TO 8000 MILLISECONDS
DEFAULT:	1000 MILLISECONDS
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

MASTER.TIMEOUT sets the length of time that the Master port communication handler will wait for a response from a field station after a message is sent through the Master port. The optimum value for this parameter is determined by the characteristics of the communication system by which the Master port communicates with its field stations.

PEERMODE.BT.ITVL

APPLICABILITY:	DATATRAIN VIII
RANGE OF VALUES:	0 TO 32767 SECONDS
DEFAULT:	0
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

PEERMODE.BT.ITVL specifies the time in seconds between periodic transmissions of the complete indication database (bit map) of a DATATRAIN VIII peer. If a bitmap message has not been sent for the specified time, a bit map message is formatted and sent. If the bit map interval is set to 0, no periodic bit map messages are sent.

PEERMODE.TIMEOUT

APPLICABILITY:	DATATRAIN VIII
RANGE OF VALUES:	30 TO 8000ms
DEFAULT:	1000ms
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

PEERMODE.TIMEOUT specifies the time in milliseconds that a DATATRAIN VIII peer will wait for an indication message to be acknowledged. Unacknowledged messages are retried indefinitely.

QUEUE.OPTION

APPLICABILITY:	ALL GENISYS 2000 APPLICATIONS
RANGE OF VALUES:	1 or 2
DEFAULT:	2
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

The selected queue option determines how logical bit changes are processed. When the 1 queue option is selected, changes are processed in the order which they occur. When the 2 queue option is selected, 0 to 1 changes (makes) are processed first for all changes which occur in the same pass of the logic processor. The default setting (2) normally produces the best performance and should not be changed arbitrarily. See section 2.6.3, Queuing Options for more information.

RF.RETRIES

APPLICABILITY:	ARES
RANGE OF VALUES:	2 TO 6
DEFAULT:	4
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

RF.RETRYS specifies the number of times a message will be retried on an ARES RF communication link.

SERIAL.BAUD

APPLICABILITY:	SERIAL.SLAVE
RANGE OF VALUES:	75, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, BITS PER SECOND
DEFAULT:	300 BITS PER SECOND
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

SERIAL.BAUD sets the data rate to be used by the Slave serial port. The 75 BPS rate is available only for MCS-1 Slave application. This option may not be used with ARES or ATCS slave types.

SERIAL.BYTE.DLY

APPLICABILITY:	SERIAL.SLAVE (MCS-1 only)
RANGE OF VALUES:	25 to 100 MILLISECONDS
DEFAULT:	25 MILLISECONDS
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

SERIAL.BYTE.DLY sets the maximum time between received bytes in an MCS-1 protocol message. If time between bytes exceeds this limit, the bytes previously received are processed. This parameter is dictated by the characteristics of the office code unit and the communication circuit to which the MCS-1 Slave is connected.

SERIAL.CARRIER

APPLICABILITY:	SERIAL.SLAVE
RANGE OF VALUES:	“KEYED” or “CONTINUOUS”
DEFAULT:	“CONTINUOUS”
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

SERIAL.CARRIER sets the type of communication channel to be used by the Slave serial port. “CONTINUOUS” carrier is used on a communication channel with full-duplex capability while “KEYED” carrier is used on a half-duplex communication channel. “KEYED” and “CONTINUOUS” refer to the state of the carrier signal leaving the Master (office) end of the communication circuit.

SERIAL.DCD.DELAY

APPLICABILITY:	SERIAL.SLAVE (MCS-1 only)
RANGE OF VALUES:	0 to 15 BIT TIMES
DEFAULT:	0 BIT TIMES
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

SERIAL.DCD.DELAY sets the delay between the detection of received carrier (DCD) and the enabling of the serial receiver in the MCS-1 Slave application. This delay aids in the rejection of carrier “turn-on” noise. This delay must be less than the key-on delay set for the Master (office) end of the communication circuit.

SERIAL.FAILTIMER

APPLICABILITY:	SERIAL.SLAVE
RANGE OF VALUES:	10 TO 600 SECONDS
DEFAULT:	300 SECONDS
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

SERIAL.FAIL.TIMER specifies the time in seconds after which the serial communication link is declared failed when no messages addressed to any of the defined Serial Slaves have been detected. When the serial fail timer expires, the system bit SERIAL.MASTER.ON is cleared. This bit may be used by the application program to determine the current status of the serial communication port.

NOTE:

If a dualport system is initiated, SER1.MASTER.ON is used in lieu of SERIAL.MASTER.ON.

SERIAL.KEYOFF

APPLICABILITY:	SERIAL.SLAVE
RANGE OF VALUES:	0 TO 300 BIT TIMES
DEFAULT:	12
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

SERIAL.KEYOFF sets the delay between the end of transmitted data and the deassertion of request to send. The required key-off delay value is dictated by the characteristics of the communication circuit to which the Slave port is connected. Twelve bit times at the selected data rate is usually adequate.

SERIAL.KEYON

APPLICABILITY:	SERIAL.SLAVE
RANGE OF VALUES:	0 TO 300 BIT TIMES
DEFAULT:	12
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

SERIAL.KEYON sets the delay between carrier key-on (request to send) and the beginning of transmitted data. The required key-on delay value is dictated by the characteristics of the communication circuit to which the Slave port is connected. Twelve bit times at the selected data rate is usually adequate.

SERIAL.PARITY

APPLICABILITY:	SERIAL.SLAVE (Except WB&S S2)
RANGE OF VALUES:	“ODD”, “EVEN”, “NONE”
DEFAULT:	NONE
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

SERIAL.PARITY sets transmit and receive parity for the Slave Serial port. This parameter should be left at its default setting unless the Slave application requires a different setting (MCS-1 Slaves, for example, usually require “EVEN” parity).

SERIAL.STOP

APPLICABILITY:	SERIAL.SLAVE (Except WB&S S2)
RANGE OF VALUES:	1 or 2
DEFAULT:	1
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

SERIAL.STOP sets the number of stop bits to be transmitted by the Slave port. The default, “1” is normally sufficient. Transmission of two stop bits may be necessary to accommodate certain timing anomalies created by communication equipment or requirements of the Slave protocol being implemented. MCS-1, for example, requires 2 stop bits. The Slave Serial port receiver always requires only 1 stop bit regardless of the stop bit setting.

SERIAL.XMT.LIMIT

APPLICABILITY:	SERIAL.SLAVE (WB&S S2 only)
RANGE OF VALUES:	5 TO 60 SECONDS
DEFAULT:	10 SECONDS
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

SERIAL.XMT.LIMIT sets the maximum time that WBSS2 Slaves’ carrier may be on continuously. If this time limit is exceeded, the WB&S S2 Slave attempts to disable its carrier by deasserting the Data Terminal Ready (DTR) signal.

SERIAL2.BAUD

APPLICABILITY:	SERIAL.SLAVE2
RANGE OF VALUES:	75, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, BITS PER SECOND
DEFAULT:	300 BITS PER SECOND
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

SERIAL2.BAUD sets the data rate to be used by the Slave serial port 2. The 75 BPS rate is available only for MCS-1 Slave application. This option may not be used with ARES or ATCS slave types.

SERIAL2.BYTE.DLY

APPLICABILITY:	SERIAL.SLAVE2 (MCS-1 only)
RANGE OF VALUES:	25 to 100 MILLISECONDS
DEFAULT:	25 MILLISECONDS
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

SERIAL2.BYTE.DLY sets the maximum time between received bytes in an MCS-1 protocol message. If time between bytes exceeds this limit, the bytes previously received are processed. This parameter is dictated by the characteristics of the office code unit and the communication circuit to which the MCS-1 Slave is connected.

SERIAL2.CARRIER

APPLICABILITY:	SERIAL.SLAVE2
RANGE OF VALUES:	“KEYED” or “CONTINUOUS”
DEFAULT:	“CONTINUOUS”
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

SERIAL2.CARRIER sets the type of communication channel to be used by the Slave serial port 2. “CONTINUOUS” carrier is used on a communication channel with full-duplex capability while “KEYED” carrier is used on a half-duplex communication channel. “KEYED” and “CONTINUOUS” refer to the state of the carrier signal leaving the Master (office) end of the communication circuit.

SER2.DCD.DELAY

APPLICABILITY:	SERIAL.SLAVE2 (MCS-1 only)
RANGE OF VALUES:	0 to 15 BIT TIMES
DEFAULT:	0 BIT TIMES
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

SER2.DCD.DELAY sets the delay between the detection of received carrier (DCD) and the enabling of the serial receiver in the MCS-1 Slave application. This delay aids in the rejection of carrier “turn-on” noise. This delay must be less than the key-on delay set for the Master (office) end of the communication circuit.

SER2.FAILTIMER

APPLICABILITY:	SERIAL.SLAVE2
RANGE OF VALUES:	10 TO 600 SECONDS
DEFAULT:	300 SECONDS
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

SER2.FAIL.TIMER specifies the time in seconds after which the serial communication link is declared failed when no messages addressed to any of the defined Serial Slaves have been detected. When the serial fail timer expires, the system bit SER2.MASTER.ON is cleared. This bit may be used by the application program to determine the current status of the serial communication port.

SERIAL2.KEYOFF

APPLICABILITY:	SERIAL.SLAVE2
RANGE OF VALUES:	0 TO 300 BIT TIMES
DEFAULT:	12
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

SERIAL2.KEYOFF sets the delay between the end of transmitted data and the deassertion of request to send. The required key-off delay value is dictated by the characteristics of the communication circuit to which the Slave port is connected. Twelve bit times at the selected data rate is usually adequate.

SERIAL2.KEYON

APPLICABILITY:	SERIAL.SLAVE2
RANGE OF VALUES:	0 TO 300 BIT TIMES
DEFAULT:	12
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

SERIAL2.KEYON sets the delay between carrier key-on (request to send) and the beginning of transmitted data. The required key-on delay value is dictated by the characteristics of the communication circuit to which the Slave port is connected. Twelve bit times at the selected data rate is usually adequate.

SERIAL2.PARITY

APPLICABILITY:	SERIAL.SLAVE2 (Except WB&S S2)
RANGE OF VALUES:	“ODD”, “EVEN”, “NONE”
DEFAULT:	NONE
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

SERIAL2.PARITY sets transmit and receive parity for the Slave Serial port 2. This parameter should be left at its default setting unless the Slave application requires a different setting (MCS-1 Slaves, for example, usually require “EVEN” parity).

SERIAL2.STOP

APPLICABILITY:	SERIAL2.SLAVE (Except WB&S S2)
RANGE OF VALUES:	1 or 2
DEFAULT:	1
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

SERIAL2.STOP sets the number of stop bits to be transmitted by the Slave port 2. The default, “1”, is normally sufficient. Transmission of two stop bits may be necessary to accommodate certain timing anomalies created by communication equipment or requirements of the Slave protocol being implemented. MCS-1, for example, requires 2 stop bits. The Slave Serial port 2 receiver always requires only 1 stop bit regardless of the stop bit setting.

SER2.XMT.LIMIT

APPLICABILITY:	SERIAL.SLAVE2 (WB&S S2 only)
RANGE OF VALUES:	5 TO 60 SECONDS
DEFAULT:	10 SECONDS
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

SER2.XMT.LIMIT sets the maximum time that WBSS2 Slaves' carrier may be on continuously. If this time limit is exceeded, the WB&S S2 Slave attempts to disable its carrier by deasserting the Data Terminal Ready (DTR) signal.

SLAVE.PORT.BAUD

APPLICABILITY:	ARES.SLAVE or ATCS.SLAVE
RANGE OF VALUES:	300, 600, 1200, 2400, 4800, 9600, 19200, BITS PER SECOND
DEFAULT:	ARES: 2400 ATCS: 9600
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

SLAVE.PORT.BAUD sets the data rate to be used by the Slave serial port. This option is valid for ARES and ATCS slave types only.

VALIDATION

APPLICABILITY:	ALL GENISYS 2000 APPLICATIONS
RANGE OF VALUES:	"ON" or "OFF"
DEFAULT:	"ON"
REQUIRED/OPTIONAL:	OPTIONAL

DESCRIPTION:

The switch specifies whether or not all functions on a control delivery board or a Master or Slave port output image be valid (have once successfully been evaluated) before output is attempted. It is normally desirable for validation to be turned on, but specific applications may require that invalid bits carry a value of 0. See section 2.5, Validation Option for more information.

2.4.1 VAR

The VARiable section of the GENISYS 2000 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 in the INTERFACE section. The format of the VAR section is as follows:

```
VAR

    <id list>
```

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

2.4.2 TIMER

The TIMER section of the GENISYS 2000 application program is used to give distinct set (pickup) or clear (drop) delays to an internal relay or output bit. A bit not specified in a TIMER statement will set or clear instantaneously. Timer delays can be specified for each bit in the application program (except bits which are declared in any INPUT section), but only 300 bit timers may be concurrently active. Bit set and clear times should not, therefore, be indiscriminately assigned.

Delays may be specified in milliseconds (MSEC) seconds (SEC), or minutes (MIN). The permissible ranges are 50 to 9999 milliseconds, 1 to 999 seconds, and 1 to 60 minutes. All timer specifications must be integer numbers. All values are internally converted to 10 millisecond ticks for use. The accuracy of all timers, regardless of how they are specified, is the same and is no more than plus or minus 20 milliseconds. The accuracy of timers for any given application program depends on application program complexity.

The general format for defining timers is as follows:

```
TIMER
                <bit ID list>:    SET=<set delay>:  <units>   CLEAR=    <clear delay>:<units>;
```

where:

<bit ID list>	-a comma separated list of previously defined bits
<set delay>	- an integer indicating the number of time units to delay before setting the bit
<clear delay>	- an integer indicating the number of time units to delay before clearing the bit
<units>	- MSEC for milliseconds, SEC for seconds, or MIN for minutes

A specific example follows:

```
  TIMER

        BIT1, BIT2:          SET=500:MSEC           CLEAR=500:MSEC;
        BIT3:                SET=1:SEC              CLEAR=2000:MSEC;
        BIT4:                SET=1000:MSEC          CLEAR=0:SEC;
```

2.4.3 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. 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 2.4.5, Run Time System Description**.**

2.4.4 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. They are listed in Table 2-4.

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

Table 2-4. Logical Operator Symbols

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 Table 2-5.

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**

Table 2-5. ASSIGN Operators Truth Table

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 2-1.

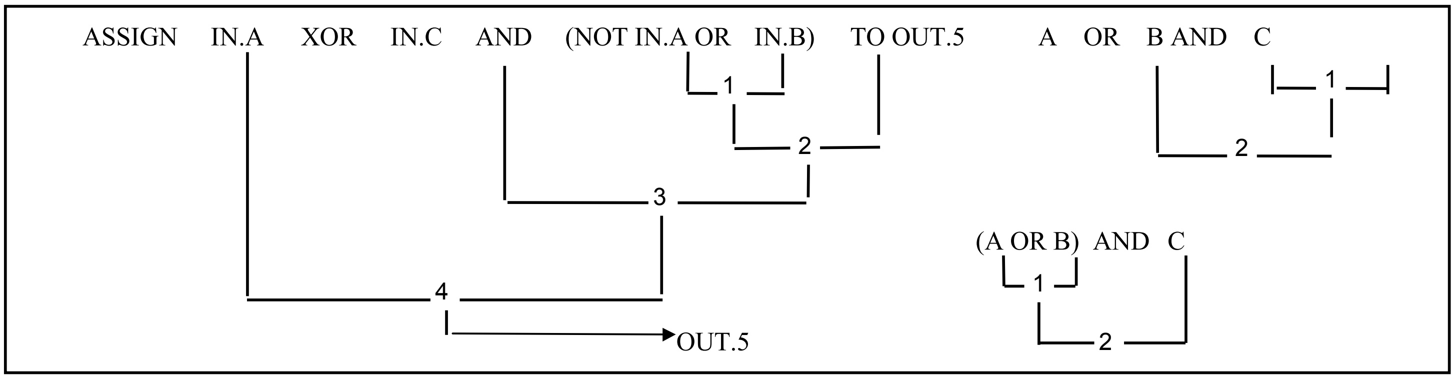


Figure 2-1. ASSIGN Operators and Order of Precedence Samples

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 Reserved Word "TO", separated by commas. For example:

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

2.4.5 Run Time System Description

The GENISYS 2000 system uses 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 value of LOCAL inputs in the table are automatically updated every 50 milliseconds. Recognition of LOCAL input change may optionally be delayed for up to 2000 milliseconds using the IND.CHANGE.DELAY configuration item. This allows filtering of unwanted transient input changes caused by external conditions such as relay contact bounce. LOCAL input bits may not be assigned a value by an ASSIGN statement as such assignment would conflict with the values assigned during the LOCAL input scan.

The LOCAL output bits are delivered in groups of 16 or 32 (depending on whether the bit is assigned to a 16 or 32 bit control delivery board). LOCAL output delivery occurs whenever any one of the bits in the group changes states as long as all bits in the group are valid or bit VALIDATION is turned off. When bit VALIDATION is turned on, LOCAL output delivery is inhibited until all bits in the group are valid. Bit VALIDATION prevents the delivery of bits which have not yet been successfully evaluated.

Input bits in the Master, Slave and DC.CODE.LINE section have their internal table value updated differently. When a GENISYS 2000 unit receives a bit through any of these ports, its received value is put in a table. It will keep that value until the bit is received again. Because the input value of these 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 bit.

The slot off of a signal request application illustrates a remote 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 signal request using a remote input bit. This request may be combined with other information to produce the actual signal request output. The green signal request is usually canceled when the appropriate track circuit becomes occupied. Note that logic may be written to clear the signal request input from the remote input port.

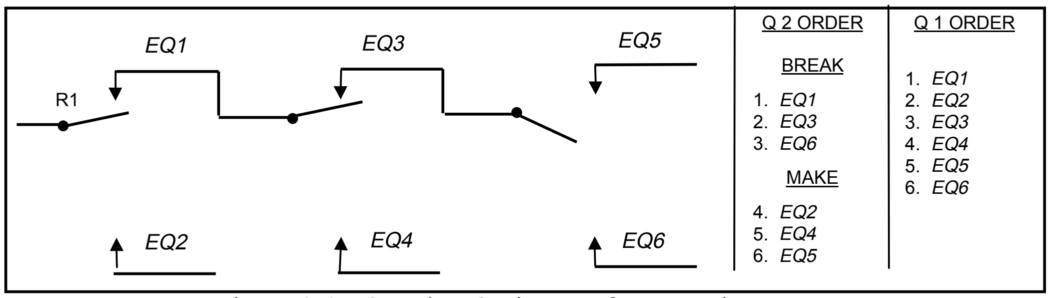
2.4.6 Logic Processing

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 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 updates (local or remote) 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.

To follow most relay-logic design 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 queuing compiler.

CONFIGURATION SECTION: QUEUE.OPTION: 2 or 1. The relay execution order in Figure 2-2 shows the operation of this switch option.



Q 2 ORDER	Q 1 ORDER
BREAK	
1. *EQ1*	1. *EQ1*
2. *EQ3*	2. *EQ2*
3. *EQ6*	3. *EQ3*
MAKE	4. *EQ4*
4. *EQ2*	5. *EQ5*
5. *EQ4*	6. *EQ6*
6. *EQ5*	

Figure 2-2. Queuing Options Reference Diagram

When QUEUE.OPTION: 1, equations are processed relay by relay, with breaks executed before makes for any given relay in the order which they occur. In Figure 2-2, 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.
With QUEUE.OPTION: 2, 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 2-2, 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.

A maximum of 4000 equations may be queued at any one time. If the 2-queue option (QUEUE.OPTION: "2 ") is used, a maximum of 2000 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 QUEUE:OPTION: "2 " option is slightly faster and should be used when possible. However, it may cause problems with some stick circuits, if this occurs consult US&S Engineering.

2.4.7 Remote Communications - Predefined Relays

When the serial I/O lines (Master or Slave) are used in GENISYS 2000 it may be necessary to condition logic processing for successful or unsuccessful completion of communications, that is 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" or other functions if communications are lost with its Master unit.

Both of these situations can be accommodated with the GENISYS 2000 system. The compiler contains predefined 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 the 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 indication message is received from a Slave unit, the associated Slave on in the Master is SET (picked). The relay will remain SET until the Master detects a communication error and exhausts retries. At this time, the relay is cleared (dropped) and will remain in this state until communications are restored with that Slave unit.

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

slave type**.MASTER.ON** (for each Slave protocol supported)

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 predefined value, it may not appear in the interface section, or be the object of an ASSIGN statement.

The MASTER.ON internal relay is SET and CLEARED differently from the SLAVE.ON. 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 a valid message addressed to it. If the communication fail timer expires before another valid message is received, the appropriate MASTER.ON is cleared.

2.4.8 Valid Bit Option

At power-on reset, the GENISYS 2000 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.

2.5 VALIDATION OPTION

The Validation option (CONFIGURATION OPTION) prevents the GENISYS 2000 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.

2.5.1 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.

2.5.2 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 2000 to a higher revision of the Executive Software, use the Validation Option (“ON”), and make sure the new configuration does not cause operational problems.

2.6 LOGIC QUEUING AND EXECUTION

2.6.1 Comparison of Hardware and Software Relay Logic

There are limits to which the GENISYS 2000 non-vital system 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 2000 which is based on a microprocessor and software, processes 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. An example is provided in Figure 2-3:

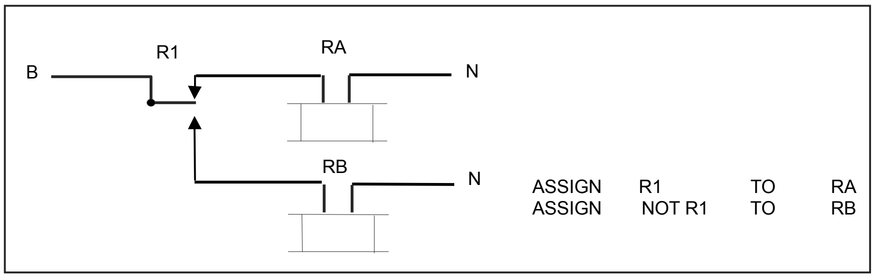


Figure 2-3. Example of Front and Back Contact Assignments

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 2000, this transfer is instantaneous. Either the front or back contact has energy applied at all times.

2.6.2 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 Software 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 picks or drops, one set of contacts is defined as the “breaks” and one set is defined as the “makes”. Those equations that are the “breaks” are executed before those that are the “makes”. When a change of state is observed by an input, internal or output bit, all logic equations that involve a contact of that bit are queued for execution, breaks before makes.

2.6.3 Queuing Options

The Executive Software has two available queuing options. These may be set in the CONFIGURATION SECTION. QUEUE. OPTIONS: "ON" sets the two separate (BREAK and MAKE) queues. The "off" option sets one queue (BREAKS before MAKES) option. The default is "ON", therefore two queues. See section 2.4.6, Logic Processing for more information.

The one-queue option has one queue in which all equations to be executed are placed, breaks before makes. The Executive Software takes equations out of the queue one at a time, starting with the first equation and continuing until the queue is empty. If any executed equation causes a change in the value of any bit, more equations may need to be executed. If so, the new equations are placed, breaks before makes, at the end of the queue after any equations that are already queued. The Executive Software continues to execute equations, one at a time, until the queue is empty. At this time, any changed outputs will be delivered to the output processors.

The two-queue option has two separate queues, one for the “breaks” and one for the “makes”. The Executive Software queues equations to be executed in the same manner as the one-queue option, except that the “breaks” go in one queue and “makes” in another queue. Equations on the “break” queue are executed before equations on the “make” queue. Any equation queued by a “break” is executed before equations queued by “makes”, even if the “make” was queued first. The difference between the two options is shown in the example in Figure 2-4:

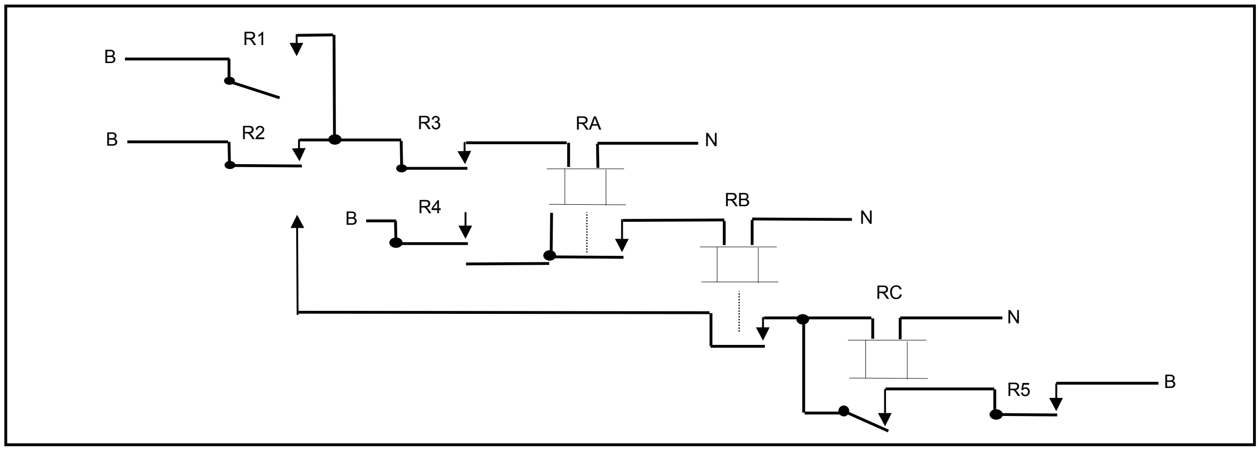


Figure 2-4. Queuing Option Example

In Figure 2-4, 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.

<u>One-queue option</u>

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.

2.6.4 Relay Models and Programming Techniques

Figure 2-5 provides sample relay circuits for the GENISYS 2000 (non-vital) programming models. The flasher relay set-up could not exist in actual relay logic, but is possible in GENISYS 2000 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;
```

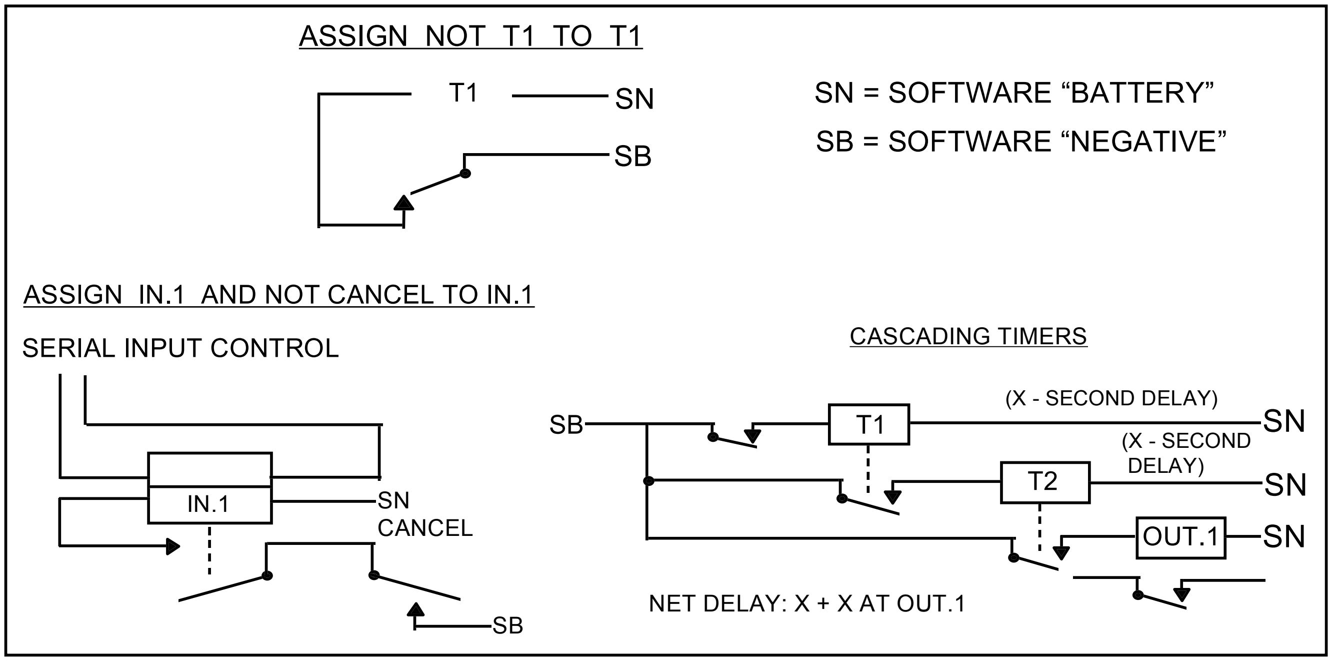


Figure 2-5. Conceptual Relay Model for GENISYS 2000 Programming

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 2-5 are conceptual models which pertain to signal control slotting. Models cannot 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):

Tl, T2: SET=5:SEC CLEAR=5:SEC

2.7 GENISYS 2000 DEVELOPMENT SYSTEM (G.D.S.2)

The GENISYS 2000 Development System (G.D.S.2) enables the user to compose, debug and load an application program into the GENISYS 2000 system hardware, Refer to Appendix A for development system components (A text editor is not available with the development system).

Figure 2-6 shows the general steps in the use of the G.D.S.2. 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.

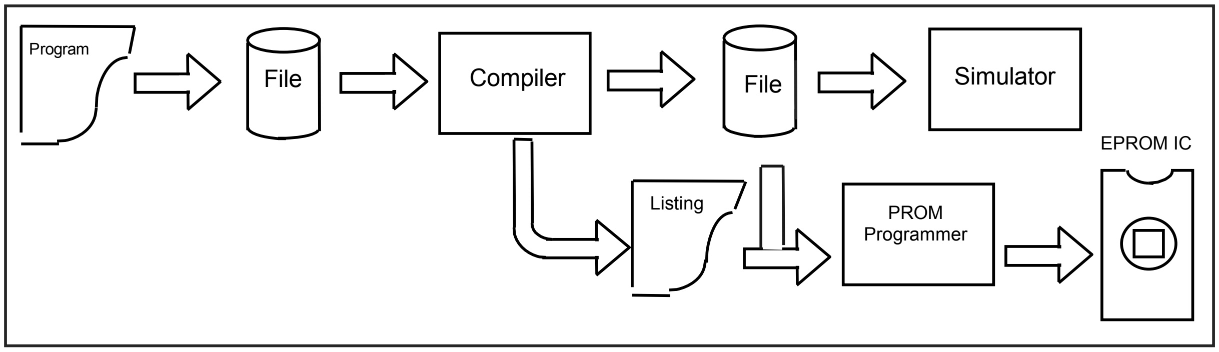


Figure 2-6. Development System Block Diagram

2.8 G.D.S.2 AVAILABLE FILES

The GENISYS 2000 Development System software uses eight file extensions. These extensions enable the user to employ the various parts of the G.D.S.2.

Extension	**File Contents**
.G2K	Source program for the application logic
.GLS	Listing file with errors produced by the compiler
.GID	Identifiers file (symbol table) produced by compiler and used by Simulator and Diagnostic Tool
.EVN	EPROM code file produced by the compiler/assembler and used by the Simulator and EPROM programmer (even bytes)
.ODD	EPROM code file produced by the compiler/assembler and used by the Simulator and EPROM programmer (odd bytes)
.GEQ	Equations generated by the Simulator
.GSI	Simulator initialization file (command file)
.GIM	Temporary intermediate file used during compilation

2.9 G.D.S.2 - COMPILER

The compiler checks and converts the application program into a code that can be processed by the Executive Software contained in separate Controller PCB EPROMs. 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 16 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 Section 6, Supplemental Data, for compiler error messages.

As the source program is processed, it is converted to a special-purpose code that 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: G2000 "name". Typing in this term invokes the batch file. This file can perform several functions, depending on how it is invoked:

GEN2000 name -If <name> .G2K appears as a file in the current directory, it is compiled; otherwise, a "file does not exist" message is displayed.

GEN2000 HELP -This displays the G.D.S.2. 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 (.EVN/.ODD) file
3. Calls the code generator
4. Calls the assembler if no errors were detected

The batch file is recommended for compiling a GENISYS 2000 program.

```
GENISYS 2000   Help File                                Version 1.0x

1. GENISYS 2000 Compiler            GENISYS 2000 name

        For Help:        GEN2000 help

Run the GENISYS 2000 compiler using the files:
        Source file    :      name.G2K      (input)
         Listing file  :      name.GLS      (output source listing)
        Symbol file    :      name.GID      (output)
        Symbol file    :      name.EVN      (output)
        Symbol file    :      name.ODD      (output)


2. GENISYS 2000 Simulator               G2SIM
   Simulate the execution of a GENISYS 2000 program.
```

2.10 G.D.S.2 - SIMULATOR

The GENISYS 2000 Simulator allows testing of a completed GENISYS 2000 application 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 stopped 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.

With the Simulator, the source program is compiled in the same manner as the program that will be loaded into the system hardware, unless the debug switch was turned off in the application by using the %D-\. The default is for this switch to always be ON and therefore always supplying the .GID file for the Simulator and diagnostic tool. The EPROM code files, with extension .EVN, and .ODD are the standard input to the Simulator. The identifier symbol file, with extension .GID, supplies relay names. Without this file, the Simulator cannot be run. In the Simulator, logic execution follows the same algorithm as the Run Time system. The Simulator supports the following two switches:

Configuration Item:	**Name**	**Comments**
QUEUE.OPTION:	Queuing Option	If QUEUE.OPTION: 2 , use two queues “break before make”, otherwise use only one queue. (QUEUE.OPTION: 1) one queue. (The default is QUEUE.OPTION: 2)
VALIDATION:	Validation Check	If VALIDATION: "ON", 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 the Valid Bit Options section. If VALIDATION: "OFF", all bits are cleared. (Default is VALIDATION: "ON".)

2.10.1 Access to Simulator

The Simulator uses the EPROM files produced by the compiler. If a program is to be debugged using the simulator, the program may not be compiled using the %$D-\ switch. The default is %$D+\ .

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. A <CR> indicates the carriage return key (or Enter).

2.10.2 Procedure

1. To run the Simulator, enter G2SIM <CR>, as shown in the Help File. The Simulator cover screen will then be displayed, showing the Simulator version.

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 total 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 2.10.3, Standard Formats. 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 2.10.6., Help Screen.

NOTE

The examples shown uses applications which were compiled with the VALIDATION: "ON". This initially flags all bits as invalid.

2.10.3 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 (1 3 7 12 19). When entering a series of consecutive items, a dash may be used for the intermediate items (1-5 7-9 11-30). Use the same procedure when listing items by name (OUT.1-OUT.3 OUT.5). If more items are requested than are available in the listing, an “invalid range" error string 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 G2SIM 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.

2.10.4 Simulator Operation

When the Simulator is executed and the EPROM tables (.ODD & .EVN) are entered, the Simulator performs the necessary housekeeping functions:

1. Initialize all bits (either CLEAR OR INVALID)
2. Reset system time
3. Initialize all queues and lists
4. Queue all equations on the Trigger List

2.10.5 Sample Program

Simulator commands are described in subsequent sections, using a sample program shown on the next page**.** Each of the available commands (Refer to section 2.10.6, Help Screen) 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.

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 their system and run the various commands as they appear.

The card slot locations of the inputs and outputs are shown in the following symbol tables.

GENISYS 2000- PROGRAM

```
GENISYS 2000 Source Listing      [ Version 1.0x ]      5-JUN-1996          Page:1
  Copyright 1993, Revised 1995, Union Switch & Signal Inc.

                 GENISYS 2000
                 PROGRAM NBR3;
                 INTERFACE
                 LOCAL
                 OUTPUT WORD:       OUT.1, OUT.2, OUT.3,
                                           SPARE, SPARE, SPARE, SPARE, SPARE,
                                           OUT.4, OUT.5, OUT.6;
                         OUTPUT WORD:      OUT.7, OUT.8;
                         INPUT WORD:              IN.A, IN.B;
                         INPUT WORD:              IN.C, IN.D;

                         VAR  STICK, T1;

                         TIMER
                           T1:          SET = 1 :SEC          CLEAR = 1 :SEC;
                           OUT.7:       SET = 1 :SEC          CLEAR = 1 :SEC;
                           OUT.8:       SET = 0:SEC           CLEAR = 500:MSEC;

                         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 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

          Errors Detected: 0
```

GENISYS 2000 SYMBOL TABLE

```
   Bit Symbol Table for "NBR3N    5-JUNE-1996                          PAGE SYM-    1
                  GENISYS 2000 Symbol Table Listing Program:NBR3

   Bit            Assigns Using              Bit Interface Information   Time      Delay
   No./Name Front / Back           Type  Station.Position                Set       Clear

   1              OUT. 1                     OUTPUT LOCAL  3   0
   2              OUT.2                      OUTPUT LOCAL  3   1
   3              OUT.3                      OUTPUT LOCAL  3   2
                  SPARE
                  SPARE
                  SPARE
                  SPARE
                  SPARE
   9              OUT4                       OUTPUT LOCAL  3   8
   10             OUT 5                      *UNASNGD LOCAL  3   9
   11             OUT.6                      OUTPUT LOCAL  3  10
   12             OUT.7                      OUTPUT LOCAL  4   0         1000:MSEC 1000:MSEC
   13             OUT.8                      OUTPUT LOCAL  4   1         0:MSEC  500:MSEC
   14             IN.A                       INPUT    LOCAL  5   0
   15             IN.B                       INPUT    LOCAL  5   1
   16             IN.C                       INPUT    LOCAL  6   0
   17             IN.D                       INPUT    LOCAL  6   1
   18             STICK                      INTERNAL
   19             T1                         INTERNAL                    1000: MSEC       1000:MSEC
   20             RESET                      *UNASGND
   21             LOG.EVENT.1                *UNASGND
   22             LOG.EVENT.2                *UNASGND
   23             SYS.CLEAR                  *UNASGND
   24             SYSERR                     STATUS BIT
   25             SYSERR.1                   STATUS BIT
   26             SYSERR.2                   STATUS BIT
   27             SYSERR.3                   STATUS BIT
   28             SYSERR.4                   STATUS BIT
   29             SYSERR.5                   STATUS BIT
   30             SYSERR.6                   STATUS BIT
   31             SYSERR.7                   STATUS BIT
   32             SYSERR.8                   STATUS BIT
   33             SYSERR.9                   STATUS BIT
   34             SYSERR.10                  STATUS BIT
   35             SERIAL.MASTER.ON           STATUS BIT
   36             ATCS.MASTER.ON             STATUS BIT
   37             DCLINE.MASTER.ON           STATUS BIT
   38             SER1,MASTER.ON             STATUS BIT
   39             SER2.MASTER.ON             STATUS BIT
   40             DUAL.PORT.ON               STATUS BIT
   41             ARES.MASTER.ON             STATUS BIT


SWITCH SETTINGS

   Master Baud Rate:            300
   Serial Baud Rate:            300
   Serial 2 Baud rate:          300
   AxxS Baud Rate:              0
   Diagnostic Baud Rate:        2400
   Master Polling Time Out:     1000 MSEC
   Control Delivery Time:       1000 MSEC
   Debug      Listing:          ON
   Symbol Table Listing:        ON
   Master Security:             OFF
   Validity Check:              ON
   Two Queue Option:            ON
```

2.10.6 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:            DlSplay        {IO}             display all values of IO boards
                    {TRiggers}                      display all equations on trigger list
                    {RElays} I(list)l               display all relays on display list
                    {Tlmers}                        display all relays on timer queue
                    VAlue (list)                    display value of relay(s)
                    REMove (lin)                    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 Ixl                     execute x number of logic equations
                    INCrement Ixl                   increment system clock x milliseconds
                    TRace Ixl                       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              Timer List:    0,    0
Command-Help                 Program: NBR3                              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 series of bits (refer also to section 3, Standard Formats).
{ }	Required argument. If not specified, it will be prompted.
file	File name. Default extensions will be added.

2.10.7 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. The left columns define input or output boards in ascending order. In keeping with the actual installation in the cardfile, the board locations are on the left side with Output or Input and its associated board number. Rows 1-8 at the top represent the output and input bit numbers, respectively, on the opto and relay boards. Remember, bit 1 on the screen is equivalent to bit 0 on the PCB. The space dividing bits 1-8 and the next group of 1-8 represent the eight-bit bytes which are 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 VALIDATION: "ON". If the validate value is VALIDATION: "OFF" then all bits equal 0. At this time, all active inputs are invalid (neither 0 nor 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" (0) or "set" (1).

2.10.8 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. Refer to Figure 2-7 for an example of the NBR3 program. The tabulation at the bottom of the 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          SIGN IN.C OR (STICK AND NOT IN.D)          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 queue value is QUEUE:OPTION: 1, only one trigger list exists.)

```
MAKE          List          BREAK               List
Line          22            Line         25
Line          23            Line         26
Line          24            Line         28
Line          26            Line         30
Line          27
                                                 29
Trigger List:   10          System Time:  00:00:00:000     Timer List:      0
Command-> Dl TR             Program: NBR3                        Screen: Triggers
```

Figure 2-7. Initial NBR3 Trigger List Screen

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

Table 2-6. Trigger List Development Table

2.10.9 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. 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, which is discussed in section 2.10.10, Remove Command. In the 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               7              |
 4   SPARE               0              |
 5   SPARE               0              |
 6   SPARE               0              |
 7   SPARE               0              |
 8   SPARE               0              |
 9   OUT.4               ?              |
10   OUT.5               ?              |
11   OUT.6               ?              |
12   OUT.7               ?              |
13   OUT.8               ?              |
14   IN.A                ?              |
15   IN.B                7              |
16   IN.C                ?
17   IN.D                ?

Trigger List:   10        System Time:  00:00:00:000        Timer List:  0
Command-> Dl TR 1-19      Program: NBR3                     Screen: Relays
```

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

2.10.10 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 that follows, 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     System Time:   00:00:00:000        Timer List:    0
 Command-> REM 4-8       Program: NBR3                       Screen: Relays
```

2.10.11 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 example below, 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 #l 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-

```
Trigger List: 10           System Time: 00:00:00:000         Timer List: 0
 Command-> lNP 1           Program: NBR3                     Screen: IO Boards

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

                     Input 1 cleared in GENISYS 2000 program
```

2.11 RELAY SET AND CLEAR COMMANDS

2.11.1 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 that follows, 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                1            |
  2   OUT.2                1            |
  3   OUT.3                0            |
  9   OUT.4                0            |
 10   OUT.5                0            |
 11   OUT.6                0            |
                                        |
            clear relay #10  OUT.5      |
            clear relay #11  OUT.6
```

2.11.2 Timer Relays

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.

The following example 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. Notice 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 1) 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            on 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                  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 (0)           |


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 (0)    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 SB 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”.

```
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 (0)          |


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.

2.11.3 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.

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 c/r

Trigger List: 10          System Time:   00:00:00:010          Timer List::  3
Command-> INC 10          Program: NBR3                        Screen: Relays

 increment time:          10 msec.
```

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 Tl, (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        Timer List :   2
Command-> INC 490       Program: NBR3                         Screen Relays
increment time          490 msec.
```

2.11.4 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 the following:

-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          Timer List:    2
Command-> Dl T1              Program:  NBR3                       Screen Relays
```

2.11.5 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 NRR3 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 2000 programs may be written so that certain logic executions do not result in a change of output. Therefore, 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                 0            |
 2    OUT.2                 ?            |
 3    OUT.3                 ?            |
 9    OUT.4                 ?            |
10    OUT.5                 ?            |
11    OUT.6                 ?            |
12    OUT.7                 ?     1000 set
13    OUT.8                 0      500 clr
14    IN.A            1                  |
15    IN.B            0                  |
16    IN.C            0                  |
17    IN.D            0                  |
18    STICK                 1
19    T1              1    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
```

2.11.6 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          System Time: 00:00:00:000       Timer List:  3
Command-> tr 3                 Program: NBR3                  Screen: Relay
clear relay #          1 OUT.1

LINE 23                ASSIGN (IN.B @ IN.A) TO OUT.2:

<CR> to execute equation
```

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 2000 shorthand notation for XOR (Q), 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.

2.11.7 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 below, 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                    0                |
  2    OUT.2                    0                |
  3    OUT.3                    0                |
  9    OUT.4                    1                |
 10    OUT.5                    ?                |
 11    OUT.6                    1                |
 12    OUT.7                    0       900 set  |
 13    OUT.8                    1       400 clr  |
 14    IN.A                     0                |
 15    IN.B                     0                |
 16    IN.C                     0                |
 17    IN.D                     0
 18    STICK                    1
 19     T1                      1       900 clr


 Trigger List:            0          System Time:    00:00:00:100            Timer List::   3
 Command-> RU 100                    Program: NBR3                           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             0            |
 2   OUT.2             0            |
 3   OUT.3             0            |
 9   OUT.4             1            |
10   OUT.5             1            |
11   OUT.6             1            |
12   OUT.              0     500 set|
13   OUT.8             1            |
14   IN.A              0            |
15   IN.B              0            |
16   IN.C              0            |
17   IN.D              0
18   STIC              1
19   T1  1                  500 clr

Trigger List:  0          System Time:   00:00:00:500          Timer List:  2
Command-> RU                      Program: NBR3                Screen: Relays
 increment time           400 msec.
```

2.11.8 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 Tl <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

```
Trigger List:  0          System Time: 00:00:01:350          Timer List:  2
Command-> VA 18 19        Program: NBR3                      Screen: IO Boards
relay #18   STICK   set
relay #19   T1      set
```

2.11.9 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-

```
      ------------------------START OF TEXT-------------------------

          di re 1-3 9-17
          inp 1

          1
          0
      Inp 2
          0
          1


        --------------------------END OF TEXT---------------------------
```

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.

2.11.10 Print Command

The Print (PR <CR>) command is used to convert a compiler EPROM tables (.EVN & .ODD files) 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, use the Quit command (QU <CR>) to leave the Simulator. Then enter NBR3.GEQ <CR> at the DOS prompt. 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 on here:

```
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 (~ IN.A) TO OUT.4

Line   26
       ASSIGN ((~IN.D) * STICK) + IN.C) TO STICK

Line   27
       ASSIGN STICK TO OUT.6;

Line   28
       ASSIGN (~Tl) TO Tl

Line   29
       ASSIGN Tl TO OUT.7

Line   30
       ASSIGN (~Tl) TO OUT.8
```

2.11.11 Reset And Quit Commands

The Reset (RES <CR >) command may be used to reset the Simulator. Using this command:

1. All bits are reset (clear or invalid)
2. Trigger and timer lists are reset
3. System time returned to 00:00:00:000
4. 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.

2.11.12 Color CRT Command

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

When using a color display, 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.

3.1 LOGIC AND TIMING OVERFLOWS

If the application program requires that more than 300 timers be concurrently active, a controlled timer queue overflow will occur. When an attempt is made to make the timer 301 active, it will force the next timer to expire prematurely. This section will continue as long as 300 timers remain active. The performance of the unit is otherwise unaffected.

3.1.1 Timing Elements

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 60 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.

3.1.2 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 the system status occurs, that causes the bit to retain its original value before the delay expires, the time delay is canceled and the bit never changes state.

4.1 LOCAL INPUT/OUTPUT (I/O) CONSIDERATIONS

4.1.1 Using Slave Units as I/O Processors

A GENISYS 2000 unit is capable of handling a maximum of 512 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 2000 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 2000 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.

4.1.2 Determining the Control Delivery Time

The Control Delivery Time is the time that the system holds a delivery 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), and the 32-bit output boards N451441-9601 and N451441-9801, control delivery time is meaningless and should be set to the lowest available value. Refer to section 2.4, Configuration Items for Application Program Parameters.

4.1.3 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 2000, but not longer than required. The Executive Software delivers control 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 delivery 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 2000 system. For example, a GENISYS 2000 system has 10 relay output boards and each of the 10 boards needs controls delivered to it. If the Control Delivery 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 2000 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.

4.1.4 Serial Communication Timing Considerations

The timing parameters associated with the serial communication between two GENISYS units are almost exclusively determined by the communication medium which is used. With two directly connected GENISYS units (units connected by an appropriate cable less than 50 feet long), serial communication may be implemented at data rates up to 19,200 bits per second. As the transmission time required for 1 data bit to travel from one unit to the other is insignificant, the elapsed time from the end of a data transmission to the beginning of the response is essentially the time required for the addressed unit to process the message. This time is typically 3 to 20 milliseconds. Therefore, when a GENISYS master is directly connected to a GENISYS slave, a no-response time out of 20 milliseconds is adequate.

As the distance between GENISYS units is increased and as additional GENISYS slave units are added, it becomes necessary to communicate using carrier equipment or modems. Depending on the level of sophistication of the equipment used, the maximum data rate available may be reduced and the end-to end communication delay time may increase. Generally, higher levels of sophistication result in higher available data rates at the expense of greater end-to-end delays. In addition, higher levels of communication equipment sophistication usually require better communication line conditioning.

Greater distances usually result in more signal processing for purposes of amplification, regeneration and signal reformatting. This, in turn, results in longer end-to-end delays. When distances of tens to hundreds of miles are involved, no-response time-outs of 70 to 400 milliseconds are usually adequate. Longer distances may require the use of higher time-out values. The no-response time out value selected must be long enough to cover the worst case communication loop delay time for circuit. The use of no-response time out values which are higher than necessary results in stable operation of the communication circuit at the cost of reduced data throughput when the circuit is disrupted by electrical noise. Note that because every master to slave transmission requires the slave unit to respond, the setting of the no-response time out has little effect on the throughput of a GENISYS communication circuit which is operating normally. It is only when slave units fail to respond that excessively long no-response time out values adversely affect data throughput. If the selected no-response time out is too short, the master unit may initiate communication with another slave before the previously addressed slave has had an opportunity to respond. This situation will result in intermittent operation of the communication circuit.

On multi-drop communication circuits, (communication circuits shared by more than 2 GENISYS units) the slave carrier must be keyed on and off to allow the circuit to be shared. When the carrier must be switched, time must be allowed for the signal to stabilize when each time carrier is turned on. This time can vary greatly depending on the type of communication equipment used. Key-on time which is measured in bit transmission times should be set to a value which allows adequate signal stabilization time. Key-off time (also measured in bit times) should be set to a value which allows time for all transmitted data bytes to pass completely through the communication circuit before carrier is turned off. For simple circuits, key-on and key-off times of 12 bit times are usually adequate through longer or more complicated circuits may require higher values. The selection of key-on and key-off delay times which are longer than necessary results in stable operation of the communication circuit with reduced data throughput. The selection of key-on and key-off delay times which are too short results in intermittent operation or failure of the communication circuit.

Finally, it is usually desirable to run GENISYS communication on a full-duplex circuit. When this is done, master to slave carrier may be left on continuously eliminating the delay required for carrier stabilization on master to slave transmissions. If this is not possible due to the characteristics of the available communication circuit, the "KEYED" carrier option may be selected. This places master and slave units in a mode where only one unit has its carrier turned on at any given time. The cost of "KEYED" (half-duplex) carrier operation is slightly higher communication delays.

It is usually possible to exactly determine the timing parameter settings required for any installation from the specifications of the communication equipment employed. Use of the exact timing parameter settings required ensures the highest possible data throughput for any hardware configuration. Contact US&S engineering for assistance on serial communication timing calculations.

5.1 GENISYS SERIAL COMMUNICATION PROTOCOL

5.1.1 General Message Format

The GENISYS protocol is a binary, byte oriented, serial, polling protocol in which all messages are framed by unique header/control and terminator bytes. It is normally transmitted and received by asynchronous serial communication controllers configured to process 8 bit characters with one start bit and one stop bit.

A typical GENISYS message is composed of a header/control character, a station address, data bytes (optional), two checksum bytes, and a terminator byte as shown in figure 5-1. Note that in the following paragraphs character (byte) values preceded by "$" are hexadecimal values.

CONTROL CHARACTER (HEADER)	STATION ADDRESS	DATA BYTE(S)	SECURITY CHECKSUM (CRC)	TERMINATOR CHARACTER

Figure 5-1. Typical GENISYS Protocol Message

5.1.1.1 Control Character

There are 13 possible header/control characters (bytes) in the GENISYS protocol ($F1 - $FE). Character $F0 is reserved for use as an "escape" character, $F6 is reserved for use as a message terminator, and $FF is not used. The basic GENISYS protocol defines 9 of the 23 available headers. Headers $F1 - $F3 are assigned to slave to master message while headers $F9 - $FE are assigned to master to slave messages.

It is possible for characters to appear in the data field which have bit patterns identical to the header/control and terminator characters. In order to preserve the uniqueness of the control and terminator characters, data bytes having values between $F0 $FF are sent as 2 byte sequences. The first byte in the sequence is always $F0 and the second byte in the sequence is (<byte value> - $F0). Whenever the character $F0 is received, it is always arithmetically added to the character which immediately follows to form the original data byte.

5.1.1.2 Station Address

In master to slave messages the station address is the address of the slave to which the message is sent. In slave to master messages the station address is the address of the slave sending the message. As there is only one master on any GENISYS protocol communication channel, and all messages are sent either to or from the master, the master requires no address. Valid station addresses are $01 to $$FF (1 to 255). The common mode message, transmitted from master to slave, may use $00 as a broadcast address.

5.1.1.3 Data Bytes

All data bytes are sent as a two byte pair. The first byte of the pair is a byte address while the second byte is the actual data. Theoretically, the range of valid data byte addresses is $00 to $DF (0 to 223 decimal). GENISYS 2000, however, normally only accepts byte addresses between $00 and $1F (and 31 decimal). Serial data bits 1 - 8 are packaged in the first byte, 9 - 16 in the second, etc. The lowest numbered bit in a data byte is the least significant bit.

5.1.1.4 Security Checksum

All GENISYS messages except the slave to master acknowledge message and the master to slave non-secure poll message include a two-byte CRC-16 checksum. The standard CRC-16 generator polynomial X(16) + X(15) + X(2) + 1 is used. This checksum is calculated before any escape characters are inserted and includes all characters in the message except the message terminator.

5.1.2 BASIC GENISYS PROTOCOL MASTER TO SLAVE MESSAGES

5.1.2.1 Common Control Message ($F9)

The common control message is used to deliver the same data to all GENISYS units on the communication channel at the same time. Common control mode must have previously been enabled for all slaves to receive the common control message by setting the common control mode flag in the slave configuration ($E0) control byte. This message is normally sent only by GENISYS master protocol handlers used in US&S computer-based CTC control machines. It is not implemented in the GENISYS 2000 master port handler. The slave units receiving the common control message are not permitted to respond.

5.1.2.2 Acknowledge and Poll Message ($FA)

The acknowledge and poll message is generated to acknowledge data successfully received from a slave. All data received from a slave must be acknowledged by the master. Unacknowledged data is repeated by the slave when subsequently addressed by the master until the data is successfully acknowledged. Valid responses include:

1. Acknowledge ($F1)
2. Data ($F2)

5.1.2.3 Poll Message ($FB)

The poll message is generated to allow the addressed slave to return any new or changed data from its database. This message has both secure and non-secure formats. The secure format, which includes a CRC-16 checksum, improves data security while the shorter non-secure form is slightly more efficient. The secure form is normally recommended. Valid response include:

1. Acknowledge ($F1)
2. Data ($F2)

5.1.2.4 Control Data Message ($FC)

The control data message is generated to pass changed or new data from the master to the slave. If the checkback control sequence is enabled the only valid response is slave checkback ($F3). If checkback is disabled valid responses included:

1. Acknowledge ($F1)
2. Data ($F2)

5.1.2.5 Indication Recall (Master Recall) Message ($FD)

The indication recall message is generated to cause the slave to respond with its entire indication database. This message is normally sent on protocol startup and any time communication is restored after an outage. It may also be sent periodically to verify the integrity of the master's database. The only valid response is a full indication data message ($F2).

5.1.2.6 Control Execute Message ($FE)

This message is generated to cause execution of previously delivered control data. It is valid only if the checkback control sequence has been previously enabled for the addressed slave and it immediately follows a valid control data message. The checkback control sequence is enabled by setting the checkback controls flag in the slave configuration control byte ($E0). Employing the checkback control sequence improves the security of data passed from master to slave. Valid responses include:

1. Acknowledge ($F1)
2. Data ($F2)

5.1.3 Basic GENISYS Protocol Slave to Master Messages

5.1.3.1 Acknowledge Message ($F1)

The acknowledge message is the default responds for a slave which receives a message from the master unit. It is sent when the slave has no data to return to the master. It is a non-secure message and includes no CRC-16 checksum. Loss or misinterpretation of a slave acknowledge message does not comprise data security.

5.1.3.2 Indication Data Message ($F2)

The indication data message is generated by a slave to deliver new or changed data to the master is in response to acknowledge, poll, control (non-checkback mode), recall or control execute (checkback mode) messages from the master.

5.1.3.3 Control Checkback Message ($F3)

A control checkback message is generated in response to a control data message received from the master when checkback control mode is enabled. The control checkback message contains all control data bytes in the control data message exactly as received from the master. This control data image is saved and executed if a control executed message is the next message form the master received by the slave. If the control checkback sequence fails, the control data is discarded and the entire sequence must be repeated.

5.1.4 GENISYS Master Protocol Drivers Operation

When the GENISYS master protocol driver is started, shortly after the GENISYS 2000 unit initialization is completed, a recall message is sent to each of the slave addresses defined in the MASTER section of the GENISYS 2000 application program. Each time the master protocol driver addresses a slave, it waits a predetermined time (the no-response time out) for a response. When the first no-response timer is aborted and a message timer is started. If a valid terminator character is received before the message timer expires, the response is immediately processed and the next communication cycle is started. A good response has a valid header/control character, is from the addressed slave, has a valid CRC-16 checksum (if applicable), has a valid terminator and a valid length.

The GENISYS master protocol driver will continue to address recall messages to each defined slave in turn until a valid response is received from each slave. As a complete indication data response is received from each slave, the slave will be marked active (SLAVE.ON.nn set), a control data message will be sent to the responding slave, and normal polling will begin.

Whenever a slave does not respond when addressed by the master or when an error is detected in the response, the station will be addressed again (usually with the same message). After at least 3 consecutive failed attempts to communicate with a slave, the slave will be marked failed (SLAVE.ON.nn cleared). Whenever retries are exhausted for a control data message, the message is discarded. When reliable communication is again established with the slave, the CURRENT control database for that slave is delivered. Failed slaves are always addressed with recall messages until they again respond correctly unless there are new control data changes to be sent. New control data changes are sent to failed slaves once. Messages sent to failed slaves are not retried.

5.1.5 GENISYS Slave Protocol Driver Operation

The GENISYS slave protocol driver constantly "listens" for commands addressed to it by the GENISYS master. A GENISYS protocol slave may never transmit a message unless it receives a properly addressed, valid message from the master. When the GENISYS slave protocol driver receives a valid message from the master it sets SERIAL.MASTER.ON indicating that communication has been established with the master. It then generates a response which is appropriate for the received message. GENISYS slaves respond to all messages addressed to them except common control messages. SERIAL.MASTER.ON remains set unless successful communication with the master has not occurred for 5 minutes (this time is user adjustable).

NOTE:

If dualport operation is initiated, SERIAL.SLAVE communication is indicated using SER1.MASTER.ON and SERIAL.SLAVE2 communication is indicated using SER2.MASTER.ON.

5.1.6 GENISYS Configuration Control Bytes

The GENISYS protocol reserves control byte addresses $E0 - $FF for configuration control purposes. At the present time, only one byte ($E0) is used. GENISYS configuration control bytes are transferred as ordinary control and indication bytes within control data ($FC) and indication data ($F2) messages.

Using the $E0 byte, the GENISYS master configures its connected slaves and they report their internal status. The bits are defined as follows:

0	- Database complete
1	- Use checkback control delivery
2	- Respond to secure poll only
3	- Enable common control message processing
4	- Reserved
5	- Reserved
6	- Reserved
7	- Reserved

5.1.7 Detailed Message Formats for Master to Slave Messages

This section describes the format for each GENISYS protocol message which is sent from master to slave. All byte values and value ranges are specified in hexadecimal. Message fields which may be repeated within a message are identified by placing square brackets [ ] around the description of the repeated field.

5.1.7.1 Common Control Message

Byte	Range	Description
1	$F9	Message header
2	$00-$FF	Slave address
	$00-$01	[Control byte address]
	$00-$FF	[Control byte data]
n-2	$00-$FF	CRC-16 low byte
n-1	$00-$FF	CRC-16 high byte
n	$F6	End of message

5.1.7.2 Acknowledge and Poll Message

Byte	Range	Description
1	$FA	Message header
2	$01-$FF	Slave address
3	$00-$FF	CRC-16 low byte
4	$00-$FF	CRC-16 high byte
5	$F6	End of message

5.1.7.3 Poll Message

Secure format:

Byte	Range	Description
1	$FB	Message header
2	$01-$FF	Slave address
3	$00-$FF	CRC-16 low byte
4	$00-$FF	CRC-16 high byte
5	$F6	End of message

Non-secure format:

Byte	Range	Description
1	$FB	Message header
2	$01-$FF	Slave address
3	$F6	End of message

5.1.7.4 Control Message

Byte	Range	Description
1	$FC	Message header
2	$01-$FF	Slave address
	$00-$1F	[Control byte address]
	$00-$FF	[Control byte data]
n-2	$00-$FF	CRC-16 low byte
n-1	$00-$FF	CRC-16 high byte
n	$F6	End of message

5.1.7.5 Recall Message

Byte	Range	Description
1	$FD	Message header
2	$01-$FF	Slave address
3	$00-$FF	CRC-16 low byte
4	$00-$FF	CRC-16 high byte
5	$F6	End of message

5.1.7.6 Control Execute Message

Byte	Range	Description
1	$FE	Message header
2	$01-$FF	Slave address
3	$00-$FF	CRC-16 low byte
4	$00-$FF	CRC-16 high byte
5	$F6	End of message

5.1.8 Detailed Message Formats for Slave to Master Messages

This section describes the format for each GENISYS protocol message which is sent from slave to master. All byte values and value ranges are specified in hexadecimal. Message fields which may be repeated within a message are identified by placing square brackets [ ] around the description of the repeated field.

5.1.8.1 Acknowledge Message

Byte	Range	Description
1	$F1	Message header
2	$01-$FF	Slave address
3	$F6	End of message

5.1.8.2 Indication Data Message

Byte	Range	Description
1	$F2	Message header
2	$01-$FF	Slave address
	$00-$1F	[Indication byte address]
	$00-$FF	[Indication byte data]
n-2	$00-$FF	CRC-16 low byte
n-1	$00-$FF	CRC-16 high byte
n	$F6	End of message

5.1.8.3 Control Data Checkback Message

Byte	Range	Description
1	$F3	Message header
2	$01-$FF	Slave address
	$00-$1F	[Control byte address]
	$00-$FF	[Control byte data]
n-2	$00-$FF	CRC-16 low byte
n-1	$00-$FF	CRC-16 high byte
n	$F6	End of message

5.1.9 Control Character Summary

Control Character	Function
$F0	Escape
$F1	Acknowledge master header
$F2	Indication data header
$F3	Control data checkback header
$F4	Reserved
$F5	Reserved
$F6	Message terminator
$F7	Reserved
$F8	Reserved
$F9	Common control data header
$FA	Acknowledge indication and poll header
$FB	Poll header
$FC	Control data header
$FD	Recall header
$FE	Execute controls header
$FF	Reserved

6.1 TOKEN AND PARSING ERROR/WARNING MESSAGES

The following tabulation lists all parsing error and warning messages that you may encounter while developing the GENISYS 2000 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 be pointing to an item past the location where the actual error occurred. You 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.	GENISYS.2000 format error	1.	Input line truncated
2.	PROGRAM statement missing	2.	Too many Identifiers
3.	INTERFACE statement missing	3.	Numeric constant > 5 digits
4.	LOCAL I/O specification expected	4.	Illegal character
5.	INPUT or OUTPUT WORD specification expected		
6.	Unexpected ID found after ";"	6.	Word more than 16 characters
7.	ADDRESS specification expected	7.	Unknown compiler switch
8.	INDICATION specification expected		
9.	CONTROL specification expected		
10.	ADDRESS MASK specification expected		
11.	CONTROL/INDICATION specification expected		
12.	SEMICOLON expected		
13.	Invalid I/O specification		
14.	Incorrect Interface format		
15.	Identifier expected		
16.	Invalid SET=OPTION		
17.	Invalid units keyword		
18.	Invalid SET/CLEAR parameter		
20.	Missing CLEAR parameter on timer		
21.	Invalid item specification		
22.	Semicolon or comma expected		
23.	Invalid TIMER		
24.	Invalid CONFIGURATION		
25.	Invalid TIMER statement		
26.	Declare missing		
27.	Invalid keyword or Begin statement missing		
28.	ASSIGN statement missing		
29.	ASSIGN or END expected		
30.	Invalid expression syntax		
31.	Invalid Parsing sequence		
32.	DC CODE TYPE missing		
33.	SERIAL.SLAVE expected		
34.	Invalid SERIAL.SLAVE syntax		
35.	ADDRESS specification expected		
36.	Integer INDICATION OFFSET expected		
37.	Integer CONTROL OFFSET expected		
38.	Invalid SLAVE/PEER address(HEX)		

39.	Invalid ARES/ATCS code type expected	
55.	Semicolon missing	
65.	Invalid SERIAL.FLAG	
66.	If INTEGER: semicolon is missing, if STRING:double quote is missing	
67.	Invalid ATCS.FLAG	
68.	Invalid DC.FLAG	
69.	SERIAL.FLAG or OUTPUT, INPUT expected	
70.	WIU.ADDRESS expected	
71.	DC.FLAG or INDICATION expected	
72.	Invalid ARES Flag	

6.2 SEMANTIC ERROR MESSAGES

The following is a list of semantic error messages that may appear while developing the GENISYS 2000 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 up to but not exceeding the maximum number of relay names due to the physical limit of that particular input or output board. Above, (n) specifies the limit that was exceeded. Another limit which may be exceeded has to do with the maximums associated with ASSIGNs, MASTER(s), SLAVE(s) and DC type(s).

2: SET or CLEAR delay more than 60 minutes.

The pick-up or drop delay was specified as greater than 60 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 (1) timing specification.

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

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

6: Invalid switch(s):

The user has specified a switch (%$...\ ) incorrectly. Valid switches are $D-\, Suppress Debug for simulator; $B-\, Suppress Symbol Table; $E+\, Enable Page Generator

7: LOCAL I/O section already defined:

This I/O specification was previously defined.

8: MASTER I/O section already defined.

This I/O specification was previously defined.

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 LOCAL I/O. It is illegal to assign these bits values with ASSIGN statements.

10: Serial Slave I/O section already defined.

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

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.

15: Illegal ATCS WIU.Address.

The user has defined a Slave unit with an illegal WIU.ADDRESS. Check for proper BCD format.

16: Illegal SERIAL.SLAVE station address.

The user has defined an invalid address for the SERIAL.SLAVE port of this unit. The address of the SLAVE port must have an address specified in the range: 0-255.

17: Illegal USS5XX CODELINE control/indication address.

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: < >

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.

24: Illegal Control / Indication length.

The length specification is invalid for code line specified.

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: Invalid MASTER baud rate.

Refer to MASTER baud rate range (300, 600, 1200, 2400, 4800, 9600, 19200, DEFAULT:300).

29: Invalid MASTER KEYON/KEYOFF range.

Refer to range settings (0-63 BITS, DEFAULT:0).

30: DC.CODELINE MASK format.

The codeline mask is not in LONG/SHORT or 1/0 format.

31: ILLEGAL USS5XX CODE LINE Mask.

The mask was not in the proper mask sequence for USS5XX codeline.

32: Set or Clear delay is less than 50 milliseconds.

Redefine the delay to a minimum of 50 milliseconds.

33: Invalid DC.CODE.TYPE.

The code type specified was NOT valid. Refer to range of valid code types for the GENISYS 2000.

34: Invalid DCD Delay.

The DCD specified was NOT valid. Refer to range: (0,15BITS,DEFAULT:0) VALID for SERIAL.CODE.TYPE:MCS (only).

35: Invalid MASTER.STOP value.

The MASTER.STOP value was NOT valid. Refer to range: (1,2, DEFAULT:1).

36: Invalid MASTER.PARITY value.

Refer to range: (ODD, EVEN, NONE, DEFAULT:NONE).

37: Invalid MASTER.TIMEOUT.

Refer to range: (30-8000 MSEC, DEFAULT:1000).

38: Invalid MASTER CARRIER.

Refer to range: (KEYED, CONTINUOUS, DEFAULT:CONTINUOUS).

39: Invalid MASTER.CHECKBACK.

Refer to range: (ENABLED, DISABLED, DEFAULT:DISABLED).

40: Invalid MASTER.COMMON parameter.

Refer to range: (ENABLED, DISABLED, DEFAULT:DISABLED).

41: Invalid SERIAL.BAUD rate.

Refer to range: (75*, 300, 600,1200, 2400, 4800, 9600, 19200)DEFAULT:300. *For SERIAL.CODE.TYPE:MCS (only).

42: Invalid SERIAL.KEYON/KEYOFF range.

Refer to range: (0- 300 BITS, DEFAULT:12).

43: Invalid SERIAL.STOP bit value.

Refer to range: (1,2, DEFAULT:1).

44: Invalid SERIAL.PARITY bit value.

Refer to range: (ODD, EVEN, DEFAULT:NONE).

45: Invalid SERIAL.BYTE.DELAY value.

Refer to range: (3-7 BITS, DEFAULT:5).

46: Invalid SERIAL.CARRIER parameter.

Refer to range: (KEYED, CONTINUOUS) DEFAULT:CONTINUOUS.

47: Invalid SERIAL.CODE.TYPE specified.

Refer to range: ”GENISYS”, “MCS1”, “WBSS2”, “DF1”, “GRSDT2”, “GRSDT2C”, “GRSDTP”, “GRSDTS”) (DEFAULT:GENISYS).

48: Maximum millisecond value reached, use SEC or MIN value.

49: Invalid DC.AUTO.RECALL specified.

Refer to range: (ENABLED, DISABLED,DEFAULT:ENABLED).

50: Invalid CONTROL.DELIVERY specified.

Refer to range: (10-4000MSEC, DEFAULT:50).

51: Invalid ATCS.LINK (HDLC) value.

Refer to range: (ODD NUMBERS (0xFF, DEFAULT:3).

52: Invalid QUEUE.OPTION value.

Refer to range: (1,2, DEFAULT:2).

53: Invalid VERSION value. DEFAULT:0.

Refer to range: (0-32000).

54: Invalid DC.CUTOUT.TIMER value.

Refer to range: (0-300, DEFAULT:180).

55: Invalid ATCS.CODE.TYPE: value. DEFAULT: “ATCS”

56: Invalid CONFIGURATION item specified.

Item specified was not a valid CONFIGURATION entry.

57: Invalid timer value.

Use a larger time value -> SEC or MIN. The maximum millisecond value has been reached, change units to SEC or MIN.

58: Transmit Limit Out of Range WBSS2 ONLY: (5-60SEC, Default: 10)

59: Illegal GRSK codeline

CONTROL address. Illegal GRSK control address type was detected.

60: Illegal GRSK codeline INDICATION address.

Illegal GRSK indication type was detected.

61: Invalid ATCS.GROUND.LINK value.

DEFAULT:23.

62: Invalid ATCS.MCP.LINK value.

DEFAULT:1.

63: Invalid FAIL.TIMER value.

Refer to range: (10-600SEC, DEFAULT:300).

64: Invalid INDICATION.CHANGE.DELAY value.

Refer to range: (50-2000MSEC, DEFAULT:1000).

65: Illegal WIU.Address.

The user has defined a ARES Slave unit with an illegal WIU.ADDRESS. Review ARES format.

66: Invalid ATCS number of valid digits (HEX).

The user has specified a number of address digits outside of range, 15 character string plus 1 length descriptor (HEX).

67: Invalid FLAG value.

Specified flag was not a valid SERIAL.FLAG or ATCS.FLAG, ARES.FLAG or DC.FLAG.

68: Invalid FRAME.LENGTH value.

Refer to range: (32, 48 ,64, 128, DEFAULT:32) For SERIAL.CODE.TYPE: S2 (only).

69: Invalid ARES number of valid digits (HEX).

The user has specified a number of address digits outside of range, 15 character string plus 1 length descriptor (HEX).

70: More than six (6) addresses specified.

The number of the particular slave type addresses has exceeded six (6).

71: SERIAL.SLAVE address already specified.

Redefine SERIAL.SLAVE address to another address descriptor.

72: Illegal GRSK codeline CONTROL MASK.

GRSK CONTROL MASK has illegal format.

73: Invalid ARES Ground Link Value. Default: 2 .

Refer to ARES address definition.

74: Illegal GRSK codeline INDICATION MASK.

GRSK INDICATION MASK has an illegal format.

75: More than MAXIMUM number of configuration items.

More than maximum number of configuration items have been specified.

76: More than MAXIMUM number of LOCAL I/O boards defined.

More than sixteen (16) I/O boards have been specified.

77: Invalid Diagnostic port address.

Refer to Diagnostic port address range (1-255); DEFAULT:1).

78: Invalid ARES LINK (HDLC) value.

Refer to range ODD NUMBERS (0xFF, DEFAULT:1).

79: Invalid ARES CODE

TYPE.DEFAULT:"ARES"

80: Invalid INDICATION ACKNOWLEDGE TIMEOUT value.

Refer to timeout range for slave type.

81: Invalid SLAVE PORT baud rate.

Refer to SLAVE PORT baud rate range (300, 600, 1200, 2400, 4800, 9600, DEFAULT:9600).

82: Invalid INDICATION BROADCAST INTERVAL value.

Refer to interval range (30-1800SEC, ATCS DEFAULT:60SEC; ARES DEFAULT:1500SEC).

83: Invalid SLAVE DESCRIPTOR TYPE.

Valid Slave types are ARES.SLAVE and ATCS.SLAVE.

84: SERIAL BAUD is invalid with ARES or ATCS Slave types.

SLAVE.PORT.BAUD is the ONLY valid baud declaration for ARES or ATCS designation.

85: Valid only for ARES slave type.

A declaration which is specific to ARES was made for another slave type. Review configuration item descriptions.

86: Invalid number of RF retries was entered.

This ARES configuration parameter was defined outside of the range (2 - 6).

87: Invalid KEYON value entered in the configuration section.

88: Invalid KEYOFF value entered in configuration section.

89: Invalid ARES station address, review specification.

90: Invalid ARES length character, review address parameter.

91: SET or CLeaR delay larger than maximum.

92: Invalid DATATRAIN VIII slave or peer address , review specification.

93: Invalid DATATRAIN type, see SERIAL.SLAVE types.

94: Invalid DATATRAIN peer mode timeout value (30-8000ms, DEFAULT: 1000ms).

95: Invalid DATATRAIN peer mode bitmap interval (0-32767sec; DEFAULT:0sec).

96: Invalid DF1 peer control offset (range 0-99 bytes; DEFAULT:0).

97: Invalid DF1 peer indication offset (range 0-99 bytes; DEFAULT:0).

98: Invalid DF1 MASTER.PEER address declared (range 0-255; DEFAULT:5).

99: Invalid diagnostic port baud rate (range 300-9600; DEFAULT:2400).

100: Invalid diagnostic carrier parameter (“CONTINUOUS” or “KEYED”; DEFAULT:”CONTINUOUS”).

101: Invalid diagnostic password.

Refer to DIAG.PASSWORD (ALPHANUMERIC: 1 to 8 in length).

102: Invalid diagnostic port stop bit value.

Refer to range: (1,2, DEFAULT:1).

103: Invalid diagnostic port parity designation.

Refer to range: (ODD, EVEN, DEFAULT:NONE).

6.3 CODE SYSTEM PRE-PROGRAMMED EPROMS

The following executive EPROMs are presently available:

N451800-0201	USS / ATCS EXECUTIVE
N451800-0202	GENISYS / 5XX EXECUTIVE
N451800-0203	GENISYS / GRS K SERIES EXECUTIVE
N451800-0204	MCS-1 EXECUTIVE
N451800-0205	PROTOCOL CONVERTER EXECUTIVE
N451800-0206	WB&S S2 EXECUTIVE

6.4 GENERAL EPROM PROGRAMMING INSTRUCTIONS

NOTE:

Review the documentation for the prom programmer device which was supplied by the manufacturer before initiating any prom programming.

6.4.1 GENISYS 2000 EPROM File Format Information

The beginning addresses for each EPROM both .ODD and .EVN is at $20000 hexadecimal.

The GENISYS 2000 Compiler produces two EPROM Code files (with .ODD extension and .EVN extension) which follow the industry standard Motorola Exormax format (S records). Most EPROM Programmers support this standard. There are a few items that must be addressed when using various Programmers:

1. The code (.ODD or .EVN) file contains the data for the EPROM. Since a checksum is stored in the EPROM to check the EPROM’s validity, any unused EPROM locations must be filled with all 1’s (or $FF for a byte) because this is the erased state of the EPROM.

Since the application program does not fill the entire EPROM, be certain that the default data that will be placed into the EPROM is all 1's. Most EPROM Programmers default to 'filling' unused locations with ones. However, many newer programmer default to zeros (this is to support different types of devices, such as PALs and PLDs).

2. A checksum may be provided by the programmer when the EPROM is programmed. This may not be the checksum of the entire EPROM. Some programmers may return the checksum of just the data that was programmed and not include all of the unused locations. To verify the checksum, a 'read' command may be used to read the entire EPROM. (Some programmers may have a separate checksum command).

3. The GENISYS 2000 EPROM Code File produced by the compiler is re-located to the addresses at which the EPROM occupies when installed into the GENISYS 2000 Hardware ($2000). It is imperative that the code be shifted to address $0000 in the EPROM[1] If this translation is not correct, the EPROMs will have the correct checksum (because all of the data is in the EPROM0, but the data will be in the wrong location. If this happens, when the EPROMs are installed into the unit, the unit will not come out of reset and display S02 (Application EPROM checksum error).

[1] The EPROMs used in the GENISYS 2000 hardware are 27C512 (512kbits or 64KBytes)
The following diagrams demonstrates how the data must be relocated into the EPROM:

<u>Example of data improperly relocated</u>

Example of data within the Code File (.ODD)		Application EPROM	EPROM physical address	EPROM address when installed in GENISYS 2000
S224020000C70F ...	>——>	FF (unused)	0	$20000
S22402002012B5 ...		FF	...	$20001
....		FF		
In this segment, the data		FF		
goes into addresses		...		
$20000 and $20020				
		C7	$20000	
		0F		
			...	
		12	$20020	
		B5		

Correct example of data properly relocated

Example of data within the Code File (.ODD)		Application EPROM	EPROM physical address	EPROM address when installed in GENISYS 2000
S224020000C70F ...	>——>	C7	0	$20000
S22402002012B5 ...	Transfer	0F	...	$20001
....	with an	...		
In this segment, the data	offset of	12		
goes into addresses	$20000	B5		
$20000 and $20020				
		FF (unused)	...	
		FF		
		FF		
		FF		
		...		

A.1 Development System Equipment

Description	US&S Part No.
Enhanced GENISYS 2000 Controller Board	N451441-9101
GENISYS 2000 Development System (5 1/4" disk)	N451232-0120
GENISYS 2000 Development System (3 1/2" disk)	N451232-0121
GENISYS 2000 Field Code Unit Diagnostic Tool (3 1/2” disk)	N451232-0123
Blank Application PROMs (27C512)	J715029-0596