S3016
User's Manual

Systems Engineering Associates, Inc.
14989 West 69th Avenue
Arvada, Colorado 80007 U.S.A.
Telephone: (303) 421-0484
Fax: (303) 421-8108

07/2001

S3016
User's Manual

CONTENTS

CONTENTS

CONTENTS

APPENDICES

SECTION 1
GENERAL DESCRIPTION

The S3016 is a communications CO-CPU board which provides one S3000 serial network interface port and one RS-232/RS-422 USER PORT. The S3016 is a true CO-CPU with its own processor and program/data memory which executes a user application program independent of the S3000 main processor. Typical applications include interfacing to operator interfaces (alphanumeric ASCII displays, CRT-based terminals, etc.) and providing S3000 network expansion. The S3016 can be installed in any I/O slot of the S3000 rack. In addition, any number of S3016 may be installed in one rack (up to the number of I/O slots available).

1.1 S3016 ARCHITECTURE

The S3016 consists of an S3000 bus interface section, S3000 serial network interface section, RS-232/RS-422 USER PORT, real time clock and a processor section.

The S3000 bus interface section consists of a 512 byte buffer which allows the S3000 main processor to pass information between the main processor and the S3016. Using the SYSdev system function, sfunc12, up to 256 bytes can be written to the S3016 and 256 bytes can be read from the S3016 in one command.

The S3000 serial network interface section allows the S3016 to be connected to the S3000 network. This allows up to 32 S3000/M4000 boards (S3012s, S3016s, M4010s, etc.) to communicate with one another on one network. The S3016 allows network expansion beyond the normal 32 node limit. Each S3016 added to the rack, allows 31 additional nodes (on a separate network) to be communicated to by the rack.

The USER PORT is a general purpose RS-232/RS-422 port accessed under software control of the user program.

The real time clock section provides the current time and date. The time is provided in a 24-hour format in the form: hours, minutes, and seconds. The date is provided in the form: month, day of month, and year. The real time clock is accurate to within 1 minute per month even in the absence of power to the S3016.

The processor section executes the user's application program. Processor memory consists of 24K bytes of battery-backed CMOS RAM program memory and 2K bytes of data memory. Typical program scan times are on the order of 0.6 milliseconds per 1K byte program memory.

SECTION 1
GENERAL DESCRIPTION

1.2 PROGRAM DEVELOPMENT

Programming is implemented using SYSdev, an IBM PC or compatible software package, that allows the user to create, document, and compile the user application program, as well as directly interface with the S3016 for program download and online monitoring. The program is developed off-line, compiled, then downloaded into the S3016. SYSdev allows the S3016 to be programmed in a combination of languages: Ladder, High-level (subset of C) and Assembly (MCS-51).

1.3 INTERFACE PORTS

The S3016 contains three interface ports: the PROGramming PORT, USER PORT, and the Serial Network Comm Port.

PROG PORT: The PROG PORT is an RS-232 port dedicated for online monitoring and program download when connected to an IBM PC or compatible running SYSdev.

USER PORT: The USER PORT is available as a general RS-232/RS-422 port for use as defined by the user. Under software control of the user application program, communications to any other RS-232/RS-422 based device can be established. Typical applications are communications to operator workstations or ASCII displays for system status or data acquisition.

Serial Network: The S3000-N1 network is a high speed (344KBPS), twisted pair, serial network configured in a master/slave topology. Communications between the S3016 and other S3000/M4000 boards (S3012, S3016, M4010, etc.) on the network is controlled via commands in the user application program resident in the master S3000 board. Data is transferred over the network using the sfunc13 system function in the master S3016 or S3012.

1.4 FAULT DETECTION/DIAGNOSTICS

Internal to the S3016 are a series of comprehensive fault detection routines which verify the proper operation of the S3016 at all times. Each detected fault has a corresponding fault code which can be viewed using SYSdev, providing a description of the fault and recommended corrective action. In addition to the fault detection routines, a hardware confidence test is resident in the S3016. This test is the same test used at the factory to verify proper hardware operation and provides a complete hardware verification of the S3016 board. This test is initiated through SYSdev.

1.5 LED STATUS INDICATIONS

The following three status LEDs are located on the S3016 faceplate: RUN, COMM, and FLT. The definitions of these LEDs are as follows:

RUN: On steady when the S3016 is running a valid users application program. Off when an internal fault is detected or when a valid user's program has not been loaded. The RUN LED is flashed during program download and also when the S3016 hardware confidence test is executed.

COMM: This LED is flashed every time an access to the S3000 serial network is made by any S3000 board on the network. If the LED is on solid, continuous communications is occurring on the network. If the LED is off, no communications is occurring. This is not a fault LED, but simply an indication of activity on the S3000 network.

FLT: "On" when an internally detected fault has occurred in the S3016. See Section 7 for more details on the S3016 fault routine and error codes.

SECTION 1
GENERAL DESCRIPTION

(This Page Intentionally Left Blank)

SECTION 2
PROGRAM STRUCTURE

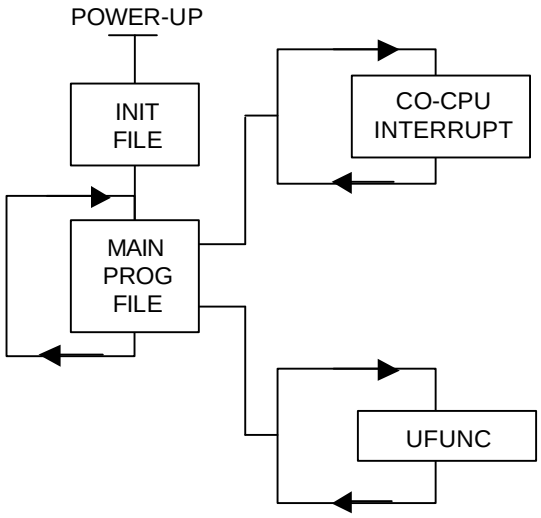
The SYSdev programming language is a combination of Ladder, High-level (subset of C) and Assembly (MCS-51). All the files shown in the following are programmed in the same language format. Each file can be written in any combination of the language types. The typical S3016 user program consists of the following files:

1) Initialization File (optional): executed once at power up.

2) Main Program File (required): scanned continuously.

3) CO-CPU Communications Interrupt File (optional): executed in response to an sfunc12 communications request from the S3012 main processor.

4) User Function Files (optional): up to 100 user defined subroutines which can be called from any of the above files.



Each file is executed sequentially from beginning to end. The main program file is executed (scanned) continuously unless interrupted by the CO-CPU interrupt. When this occurs, main program execution is suspended while the interrupt file is executed. At the completion of the interrupt, program execution resumes at the point in the main program where the interrupt occurred.

Each file is implemented as a series of consecutive blocks. Each block is defined as one of the three programming languages: Ladder, High-level or Assembly. Blocks of the different languages can be intermixed as necessary within the file.

See the SYSdev Programming Manual for more details on the typical program structure.

SECTION 2
PROGRAM STRUCTURE

(This Page Intentionally Left Blank)

SECTION 3
SYSTEM CONFIGURATION

The system configuration is used to set the following S3016 parameters: S3000 serial network baud rate, USER PORT baud rate, USER PORT parity, number of USER PORT stop bits, and to enable or disable the CO-CPU communications interrupt.

3.1 TARGET BOARD

This is used to select the board that the program will be loaded into. For the S3016, this parameter is always set to S3016. Selecting a specific board, informs the compiler to enable the appropriate features of SYSdev for the specific board.

3.2 NETWORK BAUD RATE

Three serial network baud rates are available: 344KBPS (bits per second), 229KBPS, or 106KBPS.

Note: All the boards connected on the network must be set to the same baud rate, otherwise, a communications error will occur. For the most part, the baud rate is set as a function of the total network distance. The longer the total network, the slower the baud rate. As a general rule, the baud rate can be set as follows: 344KBPS for network distance of 1000 feet or less; 229KBPS for 2000 feet or less; and 106KBPS for 4000 feet or less.

3.3 USER PORT BAUD RATE

The USER PORT baud rate can be set to any of the following baud rates: 300, 600, 1200, 2400, 4800 or 9600. The USER PORT baud rate must match the baud rate of the device connected to the USER PORT. The default baud rate is 9600.

3.4 USER PORT PARITY

The USER PORT parity can be set to NONE, ODD, or EVEN. As with the USER PORT baud rate, the parity must be set to match the device connected to the USER PORT. The default parity is NONE.

3.5 USER PORT STOP BITS

The number of stop bits for the USER PORT can be set to one or two.

Note: The USER PORT always uses one start bit and eight data bits. As with the USER PORT baud rate, the number of stop bits must be set to match the device connected to the USER PORT. The default number of stop bits is one.

3.6 CO-CPU COMMUNICATIONS INTERRUPT

If the S3012 main processor contains an sfunc12 communications call to the S3016, the S3016 CO-CPU comm interrupt must be enabled and a corresponding sfunc12 must be made in the S3016 CO-CPU comm interrupt file (see Section 6.1.1). This informs the compiler to look for and compile the CO-CPU interrupt file along with the rest of the program.

SECTION 4
VARIABLE TYPES/MEMORY MAP

4.1 VARIABLES

Three classes of variables are used in the S3016. They are: bits, bytes, and words. Bits are a single bit in width and can have a value of 0 or 1. Bytes are 8 bits in width and can have a value between 0 and 255 decimal or 0 and ffH hex. Words are 16 bits in width and can have a value of 0 to 65535 decimal or 0 to ffffH hex. All numbers (values in variables and constants) are unsigned integer values. No signed or floating point numbers are supported. Numbers can be represented as decimal or hex (suffix 'H' following number).

Three different variable types are available in the S3016: flags (F), bytes (B), words (W).

4.1.1 Flags (F):

Flags are single bit variables which are generally used as internal coils or flags in the user program. Flags can have a value of "0" or "1". The S3016 contains 104 flags.

The format of the flag variable is:

Fzzz where: zzz is a three digit flag address (000 to 103).

Note: The leading 'F' must be a capital letter and that the flag address must be three digits (include leading zeros as necessary).

Examples: F000, F012, F101, etc.

4.1.2 Bytes (B):

Byte variables are 8 bit variables used as general purpose variables in the user program. Byte variables can have a value between 0 and 255 decimal or 0 and ffH hex. Byte variables are used as arithmetic variables in the High-level language, timer/counter presets and accumulators as well as shift register bytes in the ladder language. The S3016 contains 185 ‘B’ variables.

The format of the byte variable is:

Bzzz where: zzz is the three digit byte address (032 thru 231).

Note: The leading ‘B’ must be a capital letter and that zzz must be a three digit address (include leading zeros as necessary).

Examples: B032, B150, B201, etc.

Individual bits within the byte can also be referenced by simply appending a ‘.’ followed by the bit number (0-7) to the byte address. The form of this is:

Bzzz.y where: zzz is the byte address and y is the bit (0-7).

This allows any bit in the entire data memory to be referenced just as a flag is referenced. These “byte.bit” variables can be used in ladder blocks as contact and coil variables as well as in the High-level blocks. Execution times for instructions that use bits within a byte are longer than execution times for instructions using flags. Keep this in mind when using “byte.bit” references.

Examples: B080.0, B100.7, B072.4, etc.

4.1.3 Words (W):

Word variables are 16 bit variables used as general purpose variables in the user program. Words can have a value between 0 and 65535 decimal or 0 and ffffH hex. Word variables are used as arithmetic variables in the High-level language. The S3016 contains 92 ‘W’ variables.

The format of the word variable is:

Wzzz where: zzz is the three digit word address (032 thru 230).

Note: The leading ‘W’ must be a capital letter and that zzz must be a three digit address (include leading zeros as necessary). Also, word addresses are always an even number (divisible by 2).

Examples: W034, W100, W076, etc.

4.1.4 Constants:

Constants are used as fixed numbers in High-level arithmetic and conditional statements as well as for presets in timer/counters in ladder blocks.

In High-level blocks, constants can be represented in decimal or hex. If the number is decimal, the constant is simply entered as the number to be referenced. No prefix or suffix is specified. If the number is hex, the suffix ‘H’ is added immediately following the hex number. Examples of both are:

25 (decimal)
25657 (decimal)
aeH (hex)
f000H (hex)

The hex letters (a,b,c,d,e,f) are case sensitive and must be typed as lower case letters. The hex suffix is also case sensitive and must be typed as a capital letter (H).

All constants are unsigned integers. When the variable class is byte, the range of values is 0 to 255 decimal or 0 to ffH hex. If the variable class is word, the range of values is 0 to 65535 decimal or 0 to ffffH hex.

In ladder blocks, the only constants allowed are in timer/counter presets. In this case, they are specified in decimal and preceded with the prefix '#'.

SECTION 4
VARIABLE TYPES/MEMORY MAP

4.2 DATA MEMORY MAP

The S3016 contains two distinct data memory spaces: 185 bytes of volatile (non-battery backed) data memory and 2K bytes of non-volatile (battery backed) data memory. The flag (F), byte (B) and word (W) variables, as described previously, are located in the 185 bytes of volatile data memory. The 2K bytes of non-volatile data memory can only be accessed using sfunc07 and sfunc08 (see Sections 5.2.5 and 5.2.6).

4.2.1 VOLATILE DATA MEMORY

The memory map for the S3016 volatile data memory is shown below:

Address	Valid Variable References		
0032	F000-F007	B032	W032
0033	F008-F015	B033	——
0034	F016-F023	B034	W034
0035	F024-F031	B035	——
thru	thru	thru	thru
0043	F088-F095	B043	——
0044	F096-F103	RESERVED	RESERVED
0045	RESERVED	RESERVED	RESERVED
0046	RESERVED	RESERVED	RESERVED
thru	thru	thru	thru
0055	RESERVED	RESERVED	RESERVED
0056	————	B056	W056
0057	————	B057	——
0058	————	B058	W058
thru	thru	thru	thru
0230	————	B230	W230
0231	————	B231	——

These memory locations (B032 thru B231) are not battery-backed and will not retain data at power down. At power-up or reset, these addresses are cleared.

Note: Flags F000 thru F103 are mapped into bytes B032 thru B044. Bytes B032 thru B231 are also mapped into W032 thru W230. These addresses can be referenced as any or all three of these variable types.

The flags are mapped into the bytes as shown as follows:

F000 = B032.0
F001 = B032.1
F002 = B032.2
F003 = B032.3
F004 = B032.4
F005 = B032.5
F006 = B032.6
F007 = B032.7
F008 = B033.0
F009 = B033.1
etc.

The bytes are mapped into the words with the even byte address as the low byte (lower 256 significance) of the respective word and the odd byte address as the upper byte (upper 256 significance) of the word as shown:

B032 = W032 (low byte)
B033 = W032 (high byte)

4.2.2 NON-VOLATILE (BATTERY-BACKED) DATA MEMORY

The memory map for the non-volatile (battery-backed) data memory is shown below.

Note: These memory locations are not referenced as user variables (F,B, and W) but instead are accessed using sfunc07 and sfunc08.

Address	Valid Variable References		
1800H	——	———	———
1801H	——	———	———
thru	thru	thru	thru
1fe7H	——	———	———
1fe8H	——	———	———

These variables are battery-backed and will retain data when powered down. This memory space provides a non-volatile data space for user variables such as timer/counter presets, etc. This memory space is not cleared at power-up.

Note: Addresses 1800H thru 19ffH are used as the receive/transmit buffer by sfunc12 communications system function (see Section 5.2.10).

SECTION 4
VARIABLE TYPES/MEMORY MAP

(This Page Intentionally Left Blank)

SECTION 5
PROGRAMMING REFERENCE

The following sections provide an overview of the SYSdev instruction set and the system functions available in the S3016. See the SYSdev Programming Manual for more details on the SYSdev programming language and the operation of the SYSdev software package. See appendix A for an example of an S3016 program.

5.1 INSTRUCTION SET

5.1.1 LADDER

The ladder language is generally used to implement the boolean logic of the user program. Networks of virtually any form (including nested branches) can be implemented. Ladder blocks are implemented as a 7 row X 9 column matrix. The following ladder instructions are available:

1) Contacts
   - Normally open
   - Normally closed
2) Coils
   - Standard
   - Latch
   - Unlatch
   - Inverted
3) Timers
   - 0.01 second time base
   - 0.10 second time base
   - 1.00 second time base
4) Counters
5) Shift Registers

Valid variables for contacts and coils are flags (F) or bits out of bytes (B).

Valid variables for timer/counter presets and accumulators are bytes (B). The maximum preset is 255.

Valid variables for shift registers are also bytes (B). The number of shifts per variable is 7.

5.1.2 HIGH-LEVEL ('C')

The High-level language is a subset of the 'C' programming language. High-level is used for all arithmetic, comparisons, conditional program execution, program looping, calling user functions (subroutines) and calling system functions (I/O operations). High-level blocks are implemented as a 57 row X 80 column text array.

The High-level language incorporates the following:

1) Operators:

+:	add	++:	increment
-:	subtract	—:	decrement
*:	multiply	==:	equate
/:	divide	>:	greater than
%:	remainder	>=:	greater than or equal
<<:	left shift	<:	less than
>>:	right shift	<=:	less than or equal
&:	bitwise AND	!=:	not equal
|:	bitwise OR	~:	complement
^:	bitwise EX-OR	*:	indirection (unary)
&&:	logical AND	&:	address operator
||:	logical OR	=:	equal (assignment)

2) Statements:

- program statements (equations)
- conditional program execution ("if else-if else")
- program looping ("for", "while", and "do while" loops)
- unconditional program jumping ("goto")
- user function calls ("ufuncXX" subroutines)
- system function calls ("sfuncXX" I/O operations)

5.1.3 ASSEMBLY

The Assembly language conforms to the Intel MCS-51 instruction set. The assembler syntax conforms to the UNIX system V assembler syntax.

5.2 SYSTEM FUNCTIONS

System functions provide the user with a means to perform extended I/O functions such as communication to the S3012 main processor, communication on the serial network, etc. A summary of the system functions available in the S3016 is as follows:

sfunc02: Current Time/Date Read
sfunc03: Watch Dog Timer Reset
sfunc04: ASCII String Load Command
sfunc07: General External Address Read
sfunc08: General External Address Write
sfunc09: System Fault Routine
sfunc10: USER PORT Receive
sfunc11: USER PORT Transmit
sfunc12: Intelligent I/O Block Communications
sfunc13: Serial Network Communications

System functions are entered in high-level blocks as text. Each system function has a parameter list associated with the system function call which defines such things as the address to read/write to, the number of bytes to send/receive, etc. In addition, some system functions return with an error code or function status which can be used to determine if the system function was successful, busy, etc.

5.2.1 SYSTEM FUNCTION TYPES

Two types of system functions exist in the S3016: **suspended** and **simultaneous**.

Suspended system functions actually suspend program execution while they are executed. Thus they are performed just as any other type of instruction, in order of sequence in which they occur.

Simultaneous system functions are executed simultaneously to program execution. By their nature, simultaneous system functions may take multiple main program scans to execute. These are basically “back-ground” tasks which are executed while the user application program is executing, with insignificant impact on the user program scan time.

This type of system function returns with one of four types of return values when called: "Not Busy", "Busy", "Done" or an error code representing a fault in the execution of the function. When the function is first executed, a return value of “Busy” is returned. This indicates the function is executing and is no longer available for use until it has completed. Subsequent calls to the same system function will result in a “Busy” return value until the function has completed. At that time, a call to the system function will result in either a “Done” return value or an error code value representing a failure of the function to execute. The system function is now available to execute again. See the individual system function formats following for more details on the return values and error codes pertinent to each system function.

5.2.2 sfunc02: current time/date read

System function 02 is used to read the current time and date from the real time clock embedded in the S3016. When executed, sfunc02 reads the real time clock and stores the time and date in six consecutive bytes in variable memory as follows: hours (1-24), minutes, seconds, month, day of the month, year (0-99).

Note: A 24-hour time format is used by the real time clock, thus for 9:00 am, hours = 9, for 5:00 pm, hours = 17. Also, the "year" byte will contain only the last two digits of the current year.

General form: sfunc02 (dest);

Parameters: dest = The first address of the six consecutive bytes where the current time/date will be stored.

The time and date is stored in the six consecutive bytes as follows:

byte #1: hours (1-24)
byte #2: minutes (0-59)
byte #3: seconds (0-59)
byte #4: month (1-12)
byte #5: day of month (1-31)
byte #6: year (0-99)
"dest" variable types: 'B' or indirect 'B'

Return Value: none

Type: suspended

Valid File: Initialization, Main Program, CO-CPU comm interrupt, and user functions

Example 1) sfunc02(B100);

If the above example was called at 10:23:17 pm on November 12, 1992, the following bytes will be loaded with the corresponding values:

B100 = 22 (hours = 10:00 pm)
B101 = 23 (minutes)
B102 = 17 (seconds)
B103 = 11 (month of year)
B104 = 12 (day of month)
B105 = 92 (year)

5.2.3 sfunc03: watchdog timer reset

System function 03 resets the main program watchdog timer when called. The watchdog timer normally times out if the main program scan time is longer than 100msec. This function can be used to extend this time by 100msec every time sfunc03 is called. This is desirable, for instance, if a long, intentional program loop ("for" loop, "while" loop, etc.) is executed which would exceed the normal 100msec scan time.

General form: sfunc03();

Parameters: none

Return Value: none

Type: suspended

Valid Files: Initialization, Main Program, CO-CPU comm interrupt and user functions.

5.2.4 sfunc04: ASCII string load command

System function 04 is used to convert the characters in an ASCII string to their equivalent ASCII codes and store these codes in consecutive byte addresses in variable memory (Bxxx variables)or external non-volatile memory (addresses 1800H-1fefH). System function 04 is typically used in conjunction with the User Port sfunc11 transmit system function to send ASCII strings to operator interfaces, etc.

General form: sfunc04(dest,"string");

Parameters:

dest = The address where the first ASCII character of the string will be stored. The remaining ASCII characters will be stored in consecutive byte addresses following the first byte address. Variable types: 'B' or constant 1800H-1fefH.

string = The string is from one to 60 printable characters. These characters will be converted to their equivalent ASCII codes and stored in consecutive byte addresses starting at the dest byte address.

Note: The string must be enclosed with double quotes as shown (these double quotes are not stored as part of the string, but are simply used as delimiters for the string). Any printable character can be incorporated in the string with the exception of the double quote " or back slash \. If these two characters are to be incorporated in the string, they must be preceded with the back slash (i.e. \" will incorporate the " only and \\ will incorporate just one \).

Return Value: none

Type: suspended

Valid Files: Initialization, Main Program, CO-CPU comm interrupt and user functions

Examples

1) sfunc04(B100,"example #1");

   The above example will load the following byte addresses with the corresponding ASCII codes (numbers):

   B100 = 101 (101 = ASCII code for 'e')
   B101 = 120 (120 = ASCII code for 'x')
   B102 = 97 (97 = ASCII code for 'a')
   B103 = 109 (109 = ASCII code for 'm')
   B104 = 112 (112 = ASCII code for 'p')
   B105 = 108 (108 = ASCII code for 'l')
   B106 = 101 (101 = ASCII code for 'e')
   B107 = 32 (32 = ASCII code for space)
   B108 = 35 (35 = ASCII code for '#')
   B109 = 49 (49 = ASCII code for '1')

2) sfunc04(B150,":");

   The above example will load B150 with 58 which is the ASCII code for ':'.

3) sfunc04(1a00H,"MOTOR \"on\"");

   The above example incorporates double quotes in the string and uses the back slash to designate that these double quotes are part of the string and not the string delimiters. The characters are stored in non-volatile memory as follows:

   1a00H = 77 (77 = ASCII code for 'M')
   1a01H = 79 (79 = ASCII code for 'O')
   1a02H = 84 (84 = ASCII code for 'T')
   1a03H = 79 (79 = ASCII code for 'O')
   1a04H = 82 (82 = ASCII code for 'R')
   1a05H = 32 (32 = ASCII code for space)
   1a06H = 34 (34 = ASCII code for ")
   1a07H = 111 (111 = ASCII code for 'o')
   1a08H = 110 (110 = ASCII code for 'n')
   1a09H = 34 (34 = ASCII code for ")

5.2.5 sfunc07: general external address read

System function 07 is used to read the battery-backed data memory which is not referenced as 'B' or 'W' variables. These are memory locations 1800H thru 1fefH. This system function reads one byte from the address specified.

General form: sfunc07(ext address, dest);

Parameters: ext address = The 16 bit external RAM address (1800H thru 1fefH) to be read.

dest = The variable where the value read will be stored. Variable types: B or indirect B.

Return value: sfunc07 returns with the value read from the external address.

Type: suspended

Valid Files: Initialization, Main Program and user functions.

Example: sfunc07(1900H,B100);

The above reads the non-volatile data byte address 1900H and stores the value read in B100.

5.2.6 sfunc08: general external address write

System function 08 is used to write data to the battery-backed data memory which is not referenced as ‘B’ or ‘W’ variables. These are memory locations 1800H thru 1fefH. This system function writes one byte to the address specified.

General form: sfunc08(ext address,srce);

Parameters: ext address = The 16 bit external RAM address (1800H thru 1fefH) to be written to.

srce = The variable where the value that will be written is stored. Variable types: ‘B’.

Return value: sfunc08 returns with the value written to the external address.

Type: suspended

Valid files: Initialization, Main Program and user functions.

Example: sfunc08(1805H,B101);

The above writes the data in B101 to non-volatile data byte address 1805H.

5.2.7 sfunc09: system fault routine

System function 09 provides a means for the fault routine to be called in response to a software detected fault from the user application program. The fault routine is executed as described in Section 7.1. The fault code will be set to 45H: sfunc09 generated fault.

Note: This function should only be called when a complete system shutdown is desired due to the fact that program execution will cease.

General form: sfunc09();

Parameters: none

Return value: none

Type: non-returning

Valid files: Initialization, Main Program, CO-CPU comm interrupt, and user functions.

5.2.8 sfunc10: USER PORT receive

System function 10 receives a consecutive number of bytes from the USER PORT. See Section 6.2.1 for a detailed description of the use of sfunc10.

General form: sfunc10(#rcve,dest);

Parameters: #rcve = The number of bytes to be received thru the USER PORT. Variable types: constant (1-250), 'B' or indirect 'B'.

dest = The address where the first byte received will be stored. A consecutive number of bytes (= #rcve) is received thru the USER PORT and stored in a stack starting with this address. Variable types: 'B' or indirect 'B'.

Return Values:
0 = NOT BUSY/READY
1 = BUSY
2 = DONE (receive successful)
3 = TIME OUT (bytes not received)
4 = OVERRUN ERROR
5 = PARITY ERROR
6 = FRAMING ERROR
7 = BREAK INTERRUPT

Type: simultaneous

Valid Files: Initialization and Main Program only

5.2.9 sfunc11: USER PORT transmit

System function 11 transmits a consecutive number of bytes out the USER PORT. See Section 6.2.2 for a detailed description of the use of sfunc11.

General form: sfunc11(#sent,srce);

Parameters: #sent = The number of bytes to transmit out the USER PORT. Variable types: constant (1-250), 'B' or indirect 'B'.

srce = The address where the first byte transmitted is stored. A consecutive number of bytes (= #sent) is transmitted out the USER PORT starting with this address. Variable types: 'B' or indirect 'B'.

Return Values: 0 = NOT BUSY/READY
1 = BUSY
2 = DONE (transmit successful)

Type: simultaneous

Valid Files: Initialization and Main Program only

5.2.10 sfunc12: intelligent I/O block communications

System function 12 is used to respond to an sfunc12 communications request from the S3012 main processor. See Section 6.1.1 for more details on the use of sfunc12.

General form: sfunc12 (#rcve,#sent);

Parameters: #rcve = Number of words to be received from main processor board. Variable types: constant (0-120), 'B' or indirect 'B'.

The words received from the main processor are saved in the sfunc12 receive buffer (addresses 1800H-18ffH). These addresses can be read by using sfunc07.

#sent = Number of words to be sent back to the main processor. Variable types: constant (0-120), 'B' or indirect 'B'.

The words sent to the main processor are loaded from the sfunc12 transmit buffer (addresses 1900H-19ffH). These addresses can be written to by using sfunc08.

Return Values:
0 = NOT BUSY/READY
1 = BUSY
2 = DONE (send/rcve successful)
3 = BAD REQUEST (bad comm request from main processor)
4 = BAD ACKNOWLEDGE (main processor board did not acknowledge communication)

Type: simultaneous

Valid Files: CO-CPU comm interrupt file only

5.2.11 sfunc13: serial network communications

System function 13 is used to communicate to other S3012s, S3016s or other S3000/M4000 nodes on the serial communication network. See Section 6.3 for details on the use of sfunc13 and a description of the serial network.

General form: sfunc13(slave,#sent,s_srce,s_dest,#rcve,r_srce,r_dest);

Parameters:
- slave = Address of node to communicate with. This is the network address of the slave, each slave has a unique address. Variable type: constant (1-32), 'B' or indirect 'B'.
- #sent = Number of words to send to slave. Variable types: constant (0-120), 'B' or indirect 'B'.
- s_srce = Address of send stack in master which will be sent to slave. A consecutive number of words (= #sent) will be sent to the slave starting at this address. Variable type: 'W' or indirect 'W'.
- s_dest = Starting address of stack in slave where words sent from master will be stored. Variable type: 'W' or indirect 'W'.
- #rcve = Number of words received from slave. Variable type: constant (0-120), 'B' or indirect 'B'.
- r_srce = Starting address of stack in slave where words will be sent from slave to master. Variable type: 'W' or indirect 'W'.
- r_dest = Starting address in master where words sent from slave will be stored. Variable type: 'W' or indirect 'W'.

Return values:
- 0 = NOT BUSY/READY
- 1 = BUSY
- 2 = DONE (comm with slave successful)
- 3-10H = ERROR CODE (see Section 7.4.1 for serial network communication error code descriptions).

Type: simultaneous

Valid files: Initialization and Main Program only

SECTION 6
EXTENDED I/O OPERATIONS

6.1 S3000 BUS COMMUNICATIONS

Intelligent I/O boards (here after abbreviated as CO-CPUs) are boards that contain their own processors and execute their own programs independent of the program executed in the S3012. These boards reside in I/O slots of the S3000 rack just as I/O boards reside in the rack. The bus interface of these boards is, however, different than the basic I/O boards and thus communications with CO-CPU boards is different as well.

The S3012 uses system functions 5, 6 and 12 to communicate with CO-CPU boards. This provides a free-form mechanism to pass information back and forth between the S3012 and CO-CPUs. These system functions specify a certain number of bytes (or words) to send to the CO-CPU from the S3012 and vice versa.

Two types of bus interfaces exist in CO-CPUs: the standard CO-CPU bus interface and the buffered CO-CPU bus interface. Standard bus interface CO-CPUs use sfunc05 and sfunc06 to communicate while buffered CO-CPUs use sfunc12. The standard CO-CPU does not contain a comm buffer and, therefore, handshakes data back and forth at high speed, using the suspended sfuncs 05 and 06. Buffered CO-CPUs contain an on-board bus interface buffer that allows a larger amount of data to be transferred between the S3012 and CO-CPU using the simultaneous (background) sfunc12. The S3016 contains the buffered S3000 bus interface, thus always communicates with the S3012 using sfunc12.

6.1.1 BUFFERED CO-CPU COMMUNICATIONS (sfunc12)

Systems function 12 and the buffered CO-CPU comm interface was designed to allow larger amounts of low priority data to be transferred to and from CO-CPUs without any significant impact on program scan times. For this reason, sfunc12, in the main processor, is a simultaneous sfunc that is executed simultaneously to the user application program execution. The emphasis is not on the speed of transmission, but instead on transmitting a larger amount of data with minimal impact on scan time. For this reason, it may take multiple scans of the main processor for sfunc12 to complete once it is initiated.

System function 12 communications between an S3012 and an S3016 is always initiated from the S3012. The S3016 cannot initiate the CO-CPU comm interrupt in the S3012. For this reason, the sfunc12 is always placed in the main program of the S3012 initiating comm, while the sfunc12 in the S3016 is placed in the CO-CPU comm interrupt file.

The sequence of events in an sfunc12 comm event are as follows:

1) S3012 initiates comm with S3016 using sfunc12. Program execution in S3012 continues once sfunc12 is called without waiting for a response from the S3016.

2) As the S3012 user application program execution continues, the words to be sent from the S3012 to the S3016 are transferred to the S3016 comm buffer. Both the S3012 and S3016 application programs continue to execute while this transfer takes place.

3) Once all the words to be sent to the S3016 are transferred into the comm buffer, the comm interrupt in the S3016 is initiated causing the CO-CPU interrupt file to be executed.

4) Inside the S3016 CO-CPU interrupt file, a corresponding sfunc12 is executed which reads the words, sent from the S3012, in the comm buffer and stores them in the sfunc12 receive buffer (addresses 1800H-18ffH). The S3016 then writes the words in the sfunc12 transmit buffer (addresses 1900H-19ffH) to the comm buffer that are to be sent back to the S3012. The S3016 then exits the comm interrupt file and returns to executing the main S3016 program.

5) When the S3012 detects that the S3016 comm buffer has been loaded with the words to be read from the S3016, it begins reading these values from the comm buffer. This occurs while the application program continues to execute. Once all the words have been read from the buffer, the return value for sfunc12 is set to DONE. The sfunc12 comm event is now complete.

In the above sequence, a return value of "DONE" or an "ERROR CODE" is returned when the sfunc12 in the S3016 is called. If the sfunc12 was successful, the return value is "DONE". If it was not, an error code is returned representing the nature of the fault. See Section 5.2.10 for more details on the return values.

The sfunc12 in the S3012 main program and the sfunc12 in the S3016 comm interrupt must also be in complete agreement on the number of words sent from the S3012 to the S3016 and vice versa. In other words, if the sfunc12 in the S3012 is set to send 20 words to the S3016, the sfunc12 in the S3016 comm interrupt file must be set to receive 20 words, etc. Failure to conform to this requirement, will result in an error code return value from the sfunc12.

See Section 5.2.10 for the general format, parameter list and return values of sfunc12.

Example:

1) Typical sfunc12 comm event:

S3012 main program:

```
sfunc12(7,30,W1000,20,W1100);
```

S3016 comm interrupt:

```
sfunc12(30,20);
```

Execution: The S3012 sends 30 words (W1000 thru W1058) to the comm buffer of the S3016 in slot 7. Once all the words have been loaded in the comm buffer, the comm interrupt of the S3016 is initiated, the S3016 reads the 30 words in the comm buffer and stores them in the sfunc12 receive buffer (addresses 1800H-18ffH). The S3016 then loads the comm buffer with 20 words from the sfunc12 transmit buffer (addresses 1900H-19ffh) and exits the comm interrupt file. When the S3012 detects that the comm buffer has been loaded with the 20 words from the S3016, it reads the 20 words from the comm buffer and stores them in words W1100 thru W1138.

6.2 USER PORT COMMUNICATIONS

The USER PORT is a general purpose RS-232/RS-422 port available for connection to any RS-232/RS-422 user devices. Typical applications include: S3016 connection to operator workstations, connection to IBM PC or compatibles for system data acquisition, etc. Communications through the USER PORT is achieved using sfunc10 (USER PORT read) and sfunc11 (USER PORT write). These sfuncs allow any ASCII codes from 0 to 255 to be read from or written to the port. Odd, Even, or No Parity as well as one or two stop bits can be selected by the user.

The baud rate of the user port is programmable to 300, 600, 1200, 2400, 4800, or 9600 baud. These are all set in the system configuration (see Section 3.3).

The USER PORT can be selected as either RS-232 or RS-422 (but not both). This is done via dip switch SW1 on the S3016. To set the USER PORT for RS-232, set position 1 of SW1 to "on" and position 2 of SW1 to "off". To select RS-422, set position 1 of SW1 to "off" and position2 to "on".

Note: Different pins on the USER connector are used for the RS-232 lines and RS-422 lines (see Appendix B).

6.2.1 RECEIVING THROUGH THE USER PORT (sfunc10)

Using sfunc10, from 1 to 250 consecutive bytes can be received from the USER PORT in one command. System function 10 is a simultaneous function such that once it is initiated, program execution continues without waiting for the sfunc to complete. Subsequent calls of sfunc10 result in a return value of "BUSY" until the sfunc completes (return = "DONE") or an error occurs (return = "ERROR CODE"). Since sfunc10 is a simultaneous function, the impact on the user application program scan time is negligible when an sfunc10 is executed.

The device connected to the USER PORT must send the data to the S3016 within a certain time period once sfunc10 is initiated in order to avoid a return value of "TIME OUT". In most applications, software handshaking will be required between the S3016 and user RS-232 device in order to assure the proper number of bytes is sent at the proper time.

Note: For the S3016, as the bytes are received through the USER PORT, they are stored directly into the byte addresses specified in the sfunc10 call, there is not an intermediate buffer. Therefore, the return value of sfunc10 should be monitored to determine when all the bytes have actually been received.

The parameters specified in sfunc10 are: the number of bytes to receive and the starting address of the stack to store the bytes at. See Section 5.2.8 for the general form, parameter list and return values of sfunc10.

6.2.2 TRANSMITTING THROUGH THE USER PORT (sfunc11)

Using sfunc11, from 1 to 250 consecutive bytes can be transmitted out the USER PORT in one command. System function 11 is a simultaneous function such that once it is initiated, program execution continues without waiting for the sfunc to complete. Subsequent calls of sfunc11 result in a return value of "BUSY" until the sfunc completes (return = "DONE"). Since sfunc11 is a simultaneous function, the impact on the user application program scan time is negligible when an sfunc11 is executed.

The parameters specified in sfunc11 are: the number of bytes to transmit and the starting address of the stack of bytes that will be transmitted. See Section 5.2.9 for the general form, parameter list and return values of sfunc11.

Example

1) Transmitting out the USER PORT:

   Main program:

   ```
   B060 = sfunc11(30,B120);
   ```

   Execution: The above transmits the 30 bytes between B120 and B149 out the USER PORT. The return value of sfunc11 is stored in B060. When the sfunc11 is first called, the return value will equal "BUSY" (B060=1). Subsequent calls of sfunc11 will result in a BUSY (B060=1) return value until all 30 bytes have been transmitted, at which time a return value of "DONE" (B060=2) is obtained.

   Note: Program execution is not suspended while sfunc11 is executing. Once initiated, program execution continues with subsequent calls of sfunc11 determining when all 30 bytes have actually been transmitted. The time it takes for sfunc11 to complete is a function of the selected USER PORT baud rate and the number of bytes to be transmitted.

6.3 SERIAL NETWORK COMMUNICATIONS

The serial network provides a means for multiple S3012s/S3016s to communicate with each other. The network operates in a master/slave topology. One S3012 or S3016 acts as the master and controls all communications on the network. The remaining S3012s/S3016s act as slaves and simply respond to communications requests from the master. The master can send up to 120 consecutive words and receive up to 120 consecutive words from a slave in one command. If data is to be sent from one slave to another slave, it must be done through the master (i.e. the master reads the data from the first slave and then sends it to the second slave).

Up to 32 S3012s, S3016s or other S3000/M4000 network compatible boards can be installed on one network. These 32 boards (nodes) consist of the one master and up to 31 slaves. Each node on the network is assigned a unique network address. This number is a number between 1 and 32. The network address is used to specify which slave the master is communicating to. The network address is set in the S3016 from the SYSdev Target Board Interface Menu and is downloaded directly to the S3016 from the IBM PC or compatible running SYSdev. See Section 9.2.2.

6.3.1 COMMUNICATING ON THE NETWORK (sfunc13)

System function 13 is used to execute the communications command to the slave. The parameter list of sfunc13 contains:

1) Slave network address to communicate to.
2) Number of words to be sent to slave.
3) Starting address of stack, in master, of words which will be sent to slave.
4) Starting address of stack, in slave, where the words are to be stored.
5) Number of words to be received from slave.
6) Starting address of stack, in slave, where the words will be sent from.
7) Starting address of stack, in master, where the words from the slave will be stored.

See Section 5.2.11 for a complete description of the above parameters, the general form of sfunc13 and the return values possible with sfunc13.

Note: sfunc13 is used only in the master, the slaves respond to network communications completely transparently. No commands are added to the slave programs in order to implement the serial network. Thus, only one program (the masters) in the entire network has any commands pertaining to network communications.

System function 13 is a simultaneous function such that once it is initiated, program execution continues without waiting for the sfunc to complete. Subsequent calls of sfunc13 result in a return value of "BUSY" until the sfunc completes (return = "DONE") or detects an error (return = "ERROR CODE"). See Section 7.4.1 for a description of the serial network error codes. Since sfunc13 is a simultaneous function, the impact on the user application program scan time is negligible when executed. This is also true for the responding slave. Reception and transmission on the serial network occurs concurrently with program execution, no significant increase in the scan time of the slave occurs when a slave is communicated with.

The sequence of events in a serial network comm event are as follows:

1) Master node initiates comm event by executing an sfunc13. Program execution in the master proceeds concurrently with the transmission of the words to the slave.

2) The slave receives the words from the master concurrently with its program execution. Once all words are received from the master, the slave starts transmission of the words that are to be sent from the slave to the master. This also occurs concurrently with the slave program execution.

3) The master receives the words sent from the slave concurrently with its program execution. Once all the words from the slave have been received, the subsequent call to sfunc13 results in a return value of "DONE". Until this step, calls to sfunc13 would have resulted in a "BUSY" return value.

See Section 9.2 for details on installing and wiring the network.

Example

1) Communicating from the master to a slave:

   Master S3016 main program:

   ```
   B070 = sfunc13(4,10,W080,W100,6,W060,W110);
   ```

   Execution: The above command transmits 10 words (W080 thru W098) in the master to the slave at network address 4, storing the data in W100 thru W118. The slave then transmits 6 words (W060 thru W070) to the master, storing this data at W110 thru W120. The transmission of the data was done concurrently with the program executions of both the master and the slave.

   The return value of the sfunc13 is stored in B050. Once the sfunc13 is initiated, the return value of the sfunc13 is "BUSY" (B050=1) until the transmission is complete. At that time, the return value is "DONE" (B050=2) or an error code (B050=ERROR CODE) if an error occurred in transmission.

6.4 REAL TIME CLOCK

The S3016 contains a real time clock which provides the current time and date. The time is provided in a 24-hour format (military time) in the form: hours, minutes, and seconds. In the 24-hour format, AM hours are the same as a 12-hour format (for 1:00 am, hours =1), PM hours are equal to the 12-hour format plus 12 (thus for 5:00 pm, hours =17). The date is provided in the form: month, day of month, and year. The time and date is set in the S3016 via the "Target Board Interface" menu of SYSdev. The time and date can be read from the user's application program running in the S3016 by using the sfunc02 system function.

6.4.1 SETTING THE TIME AND DATE

To set the time and date in the S3016, perform the following:

1) Connect an IBM PC or compatible loaded with SYSdev from the COM port on the PC to the "PROG PORT" on the S3016 using an RS-232 interface cable (see Appendix B).

2) Invoke SYSdev from the root directory of the drive SYSdev is loaded on by typing SYSdev <ENTER> at the DOS prompt.

3) From the SYSdev shell, select the program that is loaded in the S3016 and press <ENTER> or "F2: Edit Prog".

4) From the Main Development Menu, select "6: Target Board Interface".

5) From the Target Board Interface menu, select "8: Set Time and Date in Target Board".

6) SYSdev will read the time and date in the S3016 and display these in the "Target Board Time" and "Target Board Date" fields of the time and date menu. SYSdev will then prompt you to change the time and date, answer "Y" if the time and date is to be changed, "N" if not.

7) f "Y" was answered to the "Change Time and Date" prompt, SYSdev will then prompt for the new time. Enter the time in the form "hours:mins:secs" where hours is 1 to 24, mins is 0 to 59, and secs is 0 to 59 and then press <ENTER>.

   Note: The time variables entered are not actually loaded into the S3016 until the <ENTER> key is depressed.

8) SYSdev will not prompt for the new date. Enter the date in the form "month-day-year" where month is a number between 1 and 12, day is 1 to 31, and year is the last two digits of the year 0 to 99 and then press <ENTER>.

9) SYSdev will then prompt to change the time and date again, answer "N" to exit back to the Target Board Interface menu.

6.4.2 READING THE TIME AND DATE

System function 02 is used to read the current time and date in the user's application program running in the S3016. When executed, sfunc02 reads the real time clock and stores the time and date in six consecutive bytes in variable memory as follows: hours, minutes, seconds, month, day of month, year. See section 5.2.2 for complete details on the use of sfunc02.

SECTION 7
FAULT DETECTION

The S3016 contains comprehensive fault detection routines which verify the proper operation of the S3016 at all times.

7.1 FAULT ROUTINE EXECUTION

When a fault is detected, the following fault routine is executed:

1) User program execution is suspended.
2) "RUN" LED on S3016 is extinguished.
3) "FLT" LED on S3016 is illuminated.
4) Fault code representing the detected fault is saved in internal S3016 memory for viewing with SYSdev.

7.2 VIEWING FAULT CODES WITH SYSDEV

When a fault occurs, an IBM PC or compatible, running SYSdev, can be connected to the PROG PORT of the S3016 to view the fault codes. To view the fault codes, perform the following:

1) Connect IBM PC "COM" port to S3016 "PROG PORT" using the appropriate cable (see Appendix B).

2) Initiate SYSdev from the DOS prompt and select the user program currently loaded in the S3016.

3) From the main menu, select "Target Board Interface".

4) From the Target Board Interface menu, select "Target Board Fault Codes/Status".

The SYSdev fault display reads the fault codes from the S3016 and displays the following:

Target Board Internal Fault Codes
1) Curr Flt:
2) Last Flt:
3) CO-CPU slot:
4) Corrective action:

Communications Network Error Codes
5) Current comm error:
6) Last comm error:

Curr Flt: This is the S3016 fault code corresponding to the current detected fault along with a short description of the fault. This fault code is cleared at power-up or optionally by the user after it is displayed in the SYSdev fault display.

SECTION 7
FAULT DETECTION

Last Flt: This is the last S3016 fault code detected, shown just as the Curr Flt is shown. Unlike the Curr Flt, this fault code is not cleared at power-up. This field retains the last detected fault even when power to the S3016 is cycled. This fault code can only be cleared after it is displayed in the SYSdev fault display.

CO-CPU Slot: Not used by the S3016.

Corrective Action: This field contains a short description of the action which can be taken to correct the particular fault that was detected.

Current Comm Error: This field displays the current serial network comm error along with a short description describing the error. This field is cleared as soon as the current comm error clears.

Last Comm Error: This field displays the last error displayed in the current comm error field. Unlike the current comm error, this field retains the error code even after the error condition clears. This provides a history of the last comm error to occur.

The user has the option of clearing the fault codes when exiting the SYSdev fault display.

7.3 FAULT CODES

The following is a list of the fault codes and descriptions, as displayed in the SYSdev fault display, detected by the S3016:

Code	Description
00H	No internal fault has occurred
40H	Primary watchdog failure - timeout
42H	Cannot communicate with target board
43H	RAM battery low - program corrupted
44H	Program memory checksum error
45H	User program system fault sfunc09 call
59H	Program execution out of bounds
5AH	Address out of program memory range
5BH	Invalid interrupt
5CH	Program invalid - execution suspended
5DH	Program dump timeout - program not sent

7.3.1 WATCHDOG TIMER TIMEOUT (40H)

The watchdog timeout fault occurs when the main program scan time exceeds 100 milliseconds. The cause of this fault ranges from an unintentional infinite loop entered in the user program to a failure of the S3016 processor.

Troubleshooting:

1) Check the user program for any unintentional infinite loops. These are loops where the exit condition of the loop can never be satisfied. This can occur in "for", "while" and "do-while" loops. Also check for any "goto" jumps that cause the program to jump to a previous location in the program with no condition to stop executing the "goto".

2) Check for any loop instructions that may take longer than 100 milliseconds to execute (a large number of iterations through the loop). This situation can be corrected by calling sfunc03(), watchdog timer reset, in the loop such that the watchdog timer is reset every iteration of the loop.

3) When the 40H fault code is displayed in the SYSdev fault display, a field is displayed that reads (PC = xxxxH). The "xxxx" is a four digit hex number which equals the address (program counter) that the program was at when the watchdog timed out. If the program was in an infinite loop, this would give an indication of where the loop was. To see which block this address is in, add an assembly block at the end of the program with just the one word "test" typed into it and then compile the program. The program will compile with no errors, but will assemble with one error (no hex file created). The compiler will create a file named "assem.lst" which is the assembly list file complete with program addresses. This file can be viewed with any text editor or with the MS-DOS "type" command. The numbers in the second column from the left are the program addresses. Locate the address in this file which was displayed in the (PC = xxxxH) field. The assembly instructions for each block are headed with the block number they are in. From this, it is possible to find what block the program was at when the timeout occurred. Remove the assembly block created above to re-compile the program without error.

4) If the above all verifies, replace the S3016 and try again.

7.3.2 IBM PC TO S3016 COMMUNICATIONS FAILURE (42H)

If an attempt to read the fault codes from the S3016 results in an error code of "42H: Cannot communicate with target board", the PC cannot communicate with the S3016. This is not an internal S3016 fault, but instead a fault detected by SYSdev. The cause of this fault ranges from catastrophic failure of the S3016 to a misconnection of the PC to the S3016.

Troubleshooting:

1) Verify that the RS-232 cable is connected to "COM1" on the PC and "PROG PORT" on the S3016.

2) Verify that the RS-232 cable connecting the PC to the S3016 is wired correctly. See appendix B for the pin out of the cable.

3) If the above verifies, replace the S3016 and try again. If the problem still persists, verify the "COM1" port for proper operation (see manual from PC manufacturer).

7.3.3 INVALID PROGRAM FAULTS (5CH AND 5DH)

The "Program invalid" (5CH) fault occurs when the S3016 does not contain a valid user program. This typically occurs when a new board is installed which has never had a user program downloaded to it or after the hardware confidence test is performed, which erases the program memory. The "Program dump timeout" (5DH) fault occurs when program download to the S3016 is interrupted while program download is in progress.

Troubleshooting:

1) Dump the user program to the S3016. These faults will clear once the S3016 is loaded with a valid user program.

2) If re-loading the S3016 with the user program does not clear the fault, replace the S3016 and try again.

7.3.4 USER PROGRAM sfunc09 SYSTEM FAULT CALL (45H)

This fault code is set when the user program performs an sfunc09(); system function fault call. See the user program for the purpose of the system fault call. See Section 5.2.7 for details on sfunc09.

7.3.5 INTERNAL S3016 FAULTS (43H, 44H, 59H-5BH)

The remainder of the fault codes detected by the S3016 represent an internal failure of the S3016. These can range from the RAM battery low to invalid interrupt requests.

Troubleshooting:

1) Perform the hardware confidence test on the S3016. It may be desirable to remove the suspect S3016 from the system and to install another S3016 to get the application being controlled back up and running. The hardware confidence test can be performed in any S3000 rack. See Section 8 for details on the test.

2) Based on the results of this test, repair S3016, return S3016 for repair, or re-install S3016 in system.

7.4 SERIAL NETWORK COMMUNICATION ERRORS

Unlike the system faults, the serial network communication errors do not cause an S3016 shutdown, but instead are simply logged into the current and last comm error registers, with user program execution continuing. The current comm error represents an error that is present at the time the fault codes are viewed, while the last comm error represents the last comm error detected. The comm error codes are viewed from the SYSdev fault display, see Section 7.2 for more details.

The error codes saved in the current and last comm error registers are the same error codes returned from the sfunc13 call. The return values from the sfunc13 calls should be saved in separate 'B' variables such that when a comm error occurs, the slave that it occurred with can be determined.

SECTION 7
FAULT DETECTION

7.4.1 SERIAL NETWORK COMM ERROR CODES

The following is a list of the detected serial network communications errors:

Code	Description
00H	No network comm error
03H	More than one bus master detected
04H	sfunc13 xmitt timeout - no response
05H	sfunc13 receive timeout - no response
06H	Invalid command received from master
07H	Receive overflow
08H	Receive collision detected
09H	Receive alignment error (bad frame)
0AH	Receive CRC error
0BH	Unknown (undefined) error
0CH	Transmit no acknowledge
0DH	Transmit under run error
0EH	Transmit collision detected
0FH	Address error (outside data memory)
10H	Unexpected slave responding

7.4.2 NO RESPONSE FROM SLAVE (04H and 05H)

The no response errors occur when the master executes an sfunc13 addressed to a particular slave, but receives no response from that slave. For every execution of sfunc13, the slave will always respond to the request, even if no data is to be sent from the slave to the master. This verifies that the slave did, in fact, receive the data sent to it.

Troubleshooting:

1) Verify that the network continuity is good between the master and the slave. This can be done by observing the "COMM" LEDs on the network interface boards. Every time sfunc13 is executed, the "COMM" LEDs will flash (or be on solid for continuous communications).

2) Verify that the master and all slaves on the network are set to the correct network address they have been assigned. For each node on the network, the address must be a number between 1 and 32 and must be unique. See Section 9.2.2.

3) If the problem persists, replace the network, interface board on the slave where the problem is occurring. Next replace the network interface board on the master. If the problem persists, replace the slave S3012 or S3016.

7.4.3 SERIAL NETWORK INTEGRITY ERROR (03H, 06H-0EH, 10H)

The serial network integrity errors occur when corruption of the transmitted frame is detected. The sources of these errors range from multiple masters attempting communications on the network to excessive induced EMI on the network.

Troubleshooting:

1) Verify that only one master is communicating on the network. The master is defined as the node which is executing the sfunc13 system functions. If two nodes are executing sfunc13s simultaneously, a network collision will occur with the corresponding corruption of data.

2) Verify that the network wiring is isolated from other high voltage wiring which could induce EMI into the network. The network should be routed in a conduit separate from other wiring.

3) Replace the network interface board at the slave with which the error occurred. If the problem persists, replace the S3012 or S3016 at the slave node.

7.4.4 ADDRESS OUTSIDE RANGE (0FH)

This error occurs when an attempt to write to memory outside the data memory range occurs in either the master or slave. Verify the corresponding sfunc13 call specifies the proper data range.

SECTION 7
FAULT DETECTION

(This Page Intentionally Left Blank)

SECTION 8
HARDWARE CONFIDENCE TEST

The hardware confident test allows the S3016 hardware to be verified for proper operation. The test is resident in all S3016s and is initiated through SYSdev. The hardware confidence test is the same test used at the factory to initially test the production S3016 boards.

Note: The hardware confidence test verifies every aspect of the S3016 hardware with the exception of the S3000 bus interface, USER PORT, and the real time clock. The board should be returned to the factory for further testing if a failure of these sections is suspected.

The test is provided to the user to verify whether the S3016 hardware is functional or not. Not as a tool to repair S3016s.

If a fault is detected, the S3016 should be returned to the factory for repair. Any attempt to repair an S3016 will void the warranty.

8.1 TESTS PERFORMED

The following is a list of the tests performed by the hardware confidence test:

1) Micro controller RAM test
2) Internal Fault detection test
3) RAM memory test
4) USER PORT loop-back
5) Serial network interface test
6) RS-232 PROG PORT test

Each test performs a 100% check of the respective hardware area of the S3016 board. If a fault is detected, the test is stopped and a test fault code is displayed to indicate the nature of the hardware failure.

8.2 PERFORMING THE HARDWARE CONFIDENCE TEST

WARNING: The hardware confidence test should not be performed in an S3016 installed in a user's control system. Unpredictable operation may result while the test is being performed. The hardware confidence test should only be performed in a separate (none control system) rack.

SECTION 8
HARDWARE CONFIDENCE TEST

8.2.1 EQUIPMENT REQUIRED

In order to perform the S3016 hardware confidence test, the following is required:

1) IBM PC or compatible with SYSdev installed.

2) RS-232 interface cable to connect "COM1" on the PC to "PROG PORT" on the S3016 (see appendix B).

3) Separate S3000 (S3004CHR, S3008CHR, or S3016CHR) rack with PS3007 power supply.

8.2.2 EXECUTING THE TEST

To execute the test, perform the following steps:

1) Install S3016 to be tested in any I/O slot of the S3000 rack. Power up rack.

2) Power up PC and enter SYSdev from the DOS prompt. Enter any user program name to proceed to the SYSdev Main Development Menu.

3) Connect Interface cable to "COM1" on PC and "PROG PORT" on S3016.

4) Select "Target Board Interface" from the Main Development Menu.

5) Select "Target Board Hardware Confidence Test" from the Target Board Interface Menu.

6) Select "S3016 Confident Test" from the Confidence Test Selection Menu.

7) A prompt verifying "continuing" will now be displayed.

   Note: Proceeding with this will set the S3016 into test mode, erasing program and data memory. The user application program will have to be downloaded to the S3016 once the test is complete. Press any key to initiate test mode, "ESC" to abort the test initiation.

8) Select "Configure Test Routine" from the Test Functions Menu. Enable the tests to be executed by answering "y" to the respective prompts.

   Note: Tests 1-5 are unconditionally performed.

9) Select "Perform Test" from the Test Functions Menu and then "ENTER" to start the test. Once the test is initiated, all tests enabled will be executed repeatedly starting with test1 thru the last enabled test until any key is depressed.

If no faults are detected, the tests will continue to execute repeatedly, displaying "test passed" messages after the successful completion of each test. If a fault does occur, the test will stop and display the following:

Fault Code = XX	(test fault code and description)
Address of fault:	(memory address or I/O address where fault occurred)
Actual data at fault:	(data actually obtained at address of fault)
Expected data at fault:	(data that should have been obtained at address of fault)
Diagnostics test number:	(for factory use only)

Once a fault occurs, exit back to the Main Test Menu and re-initiate the test to reset the fault code.

Once testing is complete, exit back to the Main Test Menu and then into the previous SYSdev menus. The user application program will have to be re-loaded into the S3016, the test clears all program memory.

8.3 INTERACTIVE INTERFACE

The interactive interface menu contains selections to read the fault code (same as displayed when a fault is detected), perform diagnostics routines (for use by the factory only) and to read and write, via the PROG PORT, to any address in the S3016. In general, all these selections are for factory use and are of little significance to the user.

SECTION 8
HARDWARE CONFIDENCE TEST

(This Page Intentionally Left Blank)

SECTION 9
INSTALLATION

The following sections describe installation of the S3016 in the rack and the installation of the serial network.

CAUTION: THE INTERNAL COMPONENTS OF THE S3016 ARE SUSCEPTIBLE TO DAMAGE BY STATIC DISCHARGE, JUST AS ANY ELECTRONIC COMPONENTS ARE. WHEN HANDLING THE S3016, THE BOARD SHOULD BE HANDLED BY THE FACEPLATE ONLY AND PREFERRABLY IN A STATIC SHIELDING BAG.

9.1 INSTALLING THE S3016 IN THE RACK

The S3016 can be installed in any I/O slot of the S3000 rack. Install the S3016 as follows:

1) Select either RS-232 or RS-422 for the USER PORT based on the type of device the USER PORT will be interfaced to. This is done via dip switch SW1 located on the S3016. To set the USER PORT for RS-232, set position 1 of SW1 to "on" and position 2 of SW1 to "off". To select RS-422, set position 1 "off" and position 2 "on".

   Note: Different pins on the USER PORT connector are used for the RS-232 lines and RS-422 lines (see Appendix B).

2) Turn power to the S3000 rack off.

3) Install the S3016 in the rack by aligning the board with the card guides and sliding in until firmly seated. The board is held in the rack via captive screws located on the faceplate.

4) Connect the S3016 to the S3000 network, if used, by plugging the network field wiring connector into the network comm port, observing the proper keying of the connector.

5) Turn power to the S3000 rack on.

6) Set the network address in the S3016, if used. See Section 9.2.2.

7) Download the user application program to the S3016 using the "Download program to target board" SYSdev menu selection. See the SYSdev programming manual for more details.

To remove the S3016 from the rack, perform the following:

1) Turn power to the S3000 rack "off".

2) Pull the network field wiring connector from the comm port.

3) Loosen the captive screws located on the faceplate and gently pull the board out of the rack using the handles located on the faceplate.

Note: When installing or removing the S3016, power to the rack must be off.

SECTION 9
INSTALLATION

9.2 SERIAL NETWORK INSTALLATION

The serial network installation consists of wiring the network and setting each S3016 on the network with a unique network address. Up to 32 S3012s/S3016s can be installed on one network.

9.2.1WIRING THE SERIAL NETWORK

Refer to Figure 1 for a typical schematic of the network and for the pin outs of the network interface connectors. When wiring the network, the following rules must be followed:

1) Wire the network using Belden #9182 single-shielded twisted pair cable or an equivalent data communications cable meeting the following spec:

Wire gauge:	22AWG
Nom. impedance:	150 ohms/ft.
Nom. attenuation at 1 MHZ:	0.004 db/ft.
Twisted pair, single-shielded	

2) The total wire length of the network cannot exceed 1,000 ft. if 344KBPS is selected, 2,000 ft. at 229KBPS, and 4,000 ft. at 106KBPS.

3) The maximum number of nodes connected to one network is limited to 32 nodes.

4) The shield of the cable should be carried through the entire network, using the shield tie points on the interface connectors to achieve this. The shield tie-points on the connectors are not internally tied to anything, they are strictly tie-points. One of these tie-points should then be tied to earth ground.

5) The two extreme ends of the network should be terminated with 150 ohm resistors as shown in Figure 1.

6) The network wiring should be isolated from other high voltage wiring by routing the network in a separate conduit dedicated to the network.

7) The network should be wired directly to the network comm port connectors. No intermediate terminations or splices should be used. The network should be wired in a direct connect topology as shown, not in multi-drop or cluster topologies.

   Note: The network comm interface connectors contain two sets of + and - terminals. The two sets of terminals are tied together internally on the board (+ to +, - to -) and are provided as tie-points to ease wiring. Communications across the network will continue even if one of the nodes has failed provided all the connectors are installed in their respective board. However, if a connector is pulled from its board, communications to the boards downstream will be lost (the internal tie-point will be broken). If it is desired, this situation can be avoided by wiring the connector as shown in Figure 2.

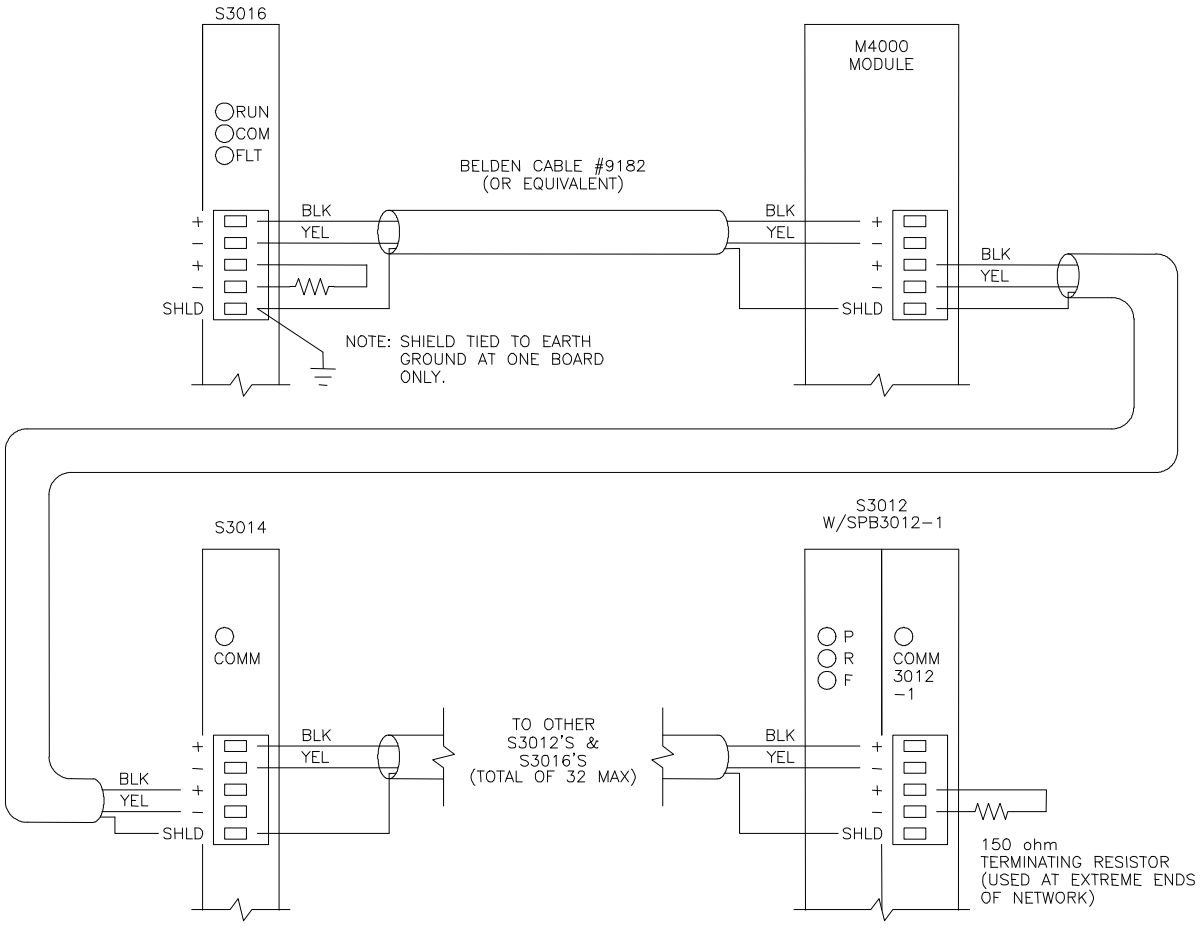


Figure 9.1 – Typical Network Wiring

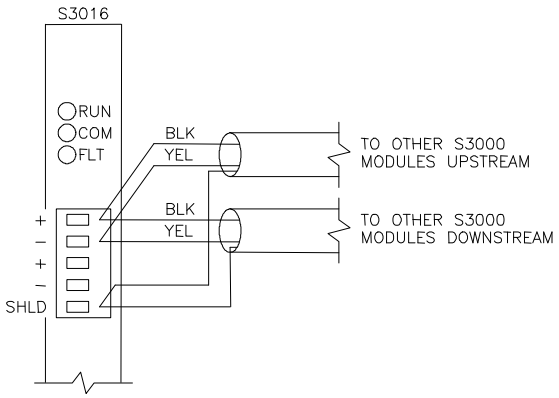


Figure 9.2 – Alternative Serial Connector Wiring

9.2.2 SETTING THE NETWORK ADDRESSES

Each S3012/S3016 on the network must be set with a unique network address between 1 and 32. This is how the S3012s/S3016s can distinguish one node from another. To set the network address for a particular S3016, perform the following:

1) Connect an IBM PC or compatible running SYSdev from "COM1" on the PC to "PROG PORT" on the S3016 using the RS-232 interface cable (see appendix B)

2) From the SYSdev Main Development Main, select "Target Board Interface".

3) From the Target Board Interface Menu, select "Target Board Network Address".

4) SYSdev will read the current network address of the S3016 and display it in the network display. If the network address is to be changed, follow the directions displayed and enter the new address.

The above steps must be done for all S3016s on the network. This is true when the network is first installed, and when a new S3016 is added or replaced (that S3016 must have the network address set in it).

SECTION 10
SPECIFICATIONS

Board Size:

Length:	9.15"
Height:	6.30"
Width:	0.80"

Processor Memory:

Program:	24K bytes battery backed CMOS RAM

Data:

Non-volatile:	2K bytes battery backed CMOS RAM

Volatile:

Flags (F):	104 bits
Bytes (B):	185 bytes
Words (W):	92 words

Interface Ports:

PROG PORT:

Type:	RS-232
Comm Rate:	9600 BAUD

USER PORT:

Type:	RS-232/RS-422
Comm Rate:	300, 600, 1200, 2400, 4800, 9600 BAUD
Start Bits:	1
Data Bits:	8
Stop Bits:	1 or 2
Parity:	None, Odd or Even

Serial Network:
S3000-N1:

Type:	RS-485
Comm Rate:	344KBPS, 229KBPS, or 106KBPS
# of Nodes (Max):	32
Isolation:	2,000 VRMS
Distance:	1,000 ft., 2,000 ft., or 4,000 ft.
Protocol:	Proprietary

Power Requirements:

Icc (+5DVC):	1.00 amps (MAX)
Icc (+12VDC):	0.10 amps (MAX)
Icc (-12VDC):	0.10 amps (MAX)

Temperature Range:

Storage:	0 to 70 degrees C
Operating:	0 to 60 degrees C

Relative Humidity:

5 to 95% (non-condensing)

SECTION 10
SPECIFICATIONS

(This Page Intentionally Left Blank)

APPENDIX A
S3016 PROGRAMMING EXAMPLES

S3016E1.LMN

```
************************************************************************
block:  1 - High-level
```

The following block is an example of communications between a master S3016 and a slave board (S3012, S3014, S3016) or module (M4010, M4020, etc.). The slave board has no program code pertaining to this communications, it responds automatically to this communications request.

The communications is implemented as follows:

1) In this example, communications is enabled when B67 is set to any value other than 0. If B67 is zero, no communications on the network is performed, if B67 is any value other than zero, network communications is performed continuously. Once enabled, the following steps are performed by the sfunc13.

2) The S3016 sends 2 words, W080 and W082, to the slave at network address 2, storing these two words a W120 and W122 respectively in the slave.

3) The S3016 then receives 2 words, W110 and W112, from the slave and stores these two words in W084 and W086 in the S3016.

   Note: sfunc13 is a simultaneous system function such that once it is initiated, program execution continues without waiting for the sfunc13 to complete. Subsequent calls of the sfunc13 will result in a return value of "BUSY" (1), "DONE" (2), or an error code (3-10H). By examining this return value, it can be determined whether it was successful (return = "DONE") or failed (return = error code).

4) In this example, once the sfunc13 is complete, it is simply called again such that communications occurs continuously and indefinitely until disabled (B67 set to 0).

```
 0:if (B67 != 0)                 /* enable serial comm? */
 1:   {                          /* yes, communicate with slave #2 */
 2:   B65 = sfunc13(2,2,W80,W120,2,W100,W84);
 3:   if (B65 != 1)              /* comm = "BUSY" */
 4:      B66 = B65;              /* save "COMM" return code */
 5:   if (B65 >= 3)              /* error code return value? */
 6:      B68 = B65;              /* yes, save error code in B68 */
 7:   }
 8:
```

```
B065  (-------)   comm   return   value
B066  (-------)   comm   return   value
B067  (-------)  serial   comm   enable
B068  (-------)   comml   error   code
W080  (-------)  display message timer
W084  (-------)  master  receive  stack
W086  (-------)  sfunc11 return   value
W100  (-------)   USER    frame   data#1
W120  (-------)   slave  receive  stack
```

APPENDIX A
S3016 PROGRAMMING EXAMPLES

```
S3016E1.LMN

************************************************************************
block:  2 - High-level

The following two blocks implement a simple ascii display driver.
The particular display driven in this example is an IEE SO3902
but this example would be typical for any ascii based display.
In this example, four different messages are displayed consecutively
for 5 seconds each repeatedly.  The driver is implements as follows:

1) When a new message is to be displayed (B90 != B91), the ascii
   characters for that message are converted to the equivalent
   ascii codes using sfunc04 and placed in consecutive variable
   locations B100 thru B139.  This is a buffer which is used as the
   frame which will be transmitted to the display out the USER PORT.

2) Once the ascii characters are converted, the entire frame is
   transmitted out the USER PORT using sfunc11. The frame is preceded
   with control codes unique to the display which clear the display
   and relocates the cursor, etc.

   Note: The sfunc11 is a simultaneous system function such that
   once it is initiated, program execution continues without waiting
   for the sfunc11 to complete. Subsequent calls of the sfunc11 will
   result in a return value of "BUSY" (1) or "DONE" (2). By examining
   this return value, it can be determined whether the sfunc11 is BUSY
   or has completed and is ready to be called again. In this particular
   driver, once sfunc11 is initiated, it is not called again until a
   new message is to be displayed.

 0:if (B90 == B91)         /* new message to display? */
 1:   goto iee_bypass;     /* no, bypass display update */
 2:
 3:/* decode message number to display and convert to ascii */
 4:if (B90 == 1)
 5:   {sfunc04(B100,"     display message #1                ");}
 6:else if (B90 == 2)
 7:   {sfunc04(B100,"     display message #2                ");}
 8:else if (B90 == 3)
 9:   {sfunc04(B100,"     display message #3                ");}
10:else if (B90 == 4)
11:   {sfunc04(B100,"     display message #4                ");}
12:else
13:   ;
14:
15:/* precede USER PORT frame with control codes */
16:B94 = 19H;                    /* multi-byte command */
17:B95 = 0cH;                    /* clear display and cursor home */
18:B96 = 0eH;                    /* invisible cursor */
19:B97 = 1bH;                    /* position cursor */
20:B98 = 00H;                    /* at 00 */
21:B99 = 11H;                    /* normal data entry mode */
22:B140 = 0dH;                   /* carriage return */
23:B86 = sfunc11(47,B94);        /* transmit frame out USER PORT */
24:if (B86 == 2)                 /* was USER PORT just finishing */
25:   B87 = sfunc11(47,B94);     /* re-initiate transmit */
26:
```

```
S3016E1.LMN

************************************************************************
block:  2 - High-level (continued)

27:iee_bypass:
28:B91 = B90;                    /* update message number */
29:
30:if (F000 == 1)                /* display new message? */
31:   {                          /* yes */
32:   ++B90;                     /* display next message */
33:   if (B90 == 5)              /* last message displayed? */
34:      B90 = 1;                /* start over with message #1 */
35:   }
36:
```

```
F000  ( (32.0))  display   new   message
B086  (-------)  sfunc11 return   value
B087  (-------)  sfunc11 return   value
B090  (-------)          message number
B091  (-------)   prev   message number
B094  (-------)   USER    frame   cmmd#1
B095  (-------)   USER    frame   cmmd#2
B096  (-------)   USER    frame   cmmd#3
B097  (-------)   USER    frame   cmmd#4
B098  (-------)   USER    frame   cmmd#5
B099  (-------)   USER    frame   cmmd#6
B100  (-------)   USER    frame   data#1
THRU
B139  (-------)   USER    frame  data#40
B140  (-------)   USER    frame    end
```

APPENDIX A
S3016 PROGRAMMING EXAMPLES

```
S3016E1.LMN

*************************************************************************
block:  3 - Ladder

    display                                                          display
      new                                                              new
    message      Timer                                               message
     F000     +---------+                                            F000
 0:+--]/[--+--|         |--+-------+-------+-------+-------+-------+--( )--
     (32.0)   | P:#00050|                                            (32.0)
              | TB:0.10 |
              | A:B080  |
              |         |
              | display |
              | message |
 1:           | timer   |
              +---------+
```

```
S3016E1.LMN

************************************************************************
block:  4 - High-level

In this block, ascii data is received from the IEE display. The
display sends two bytes out it's RS-232 port when a key is
depressed on the front of the display. The first byte indicates
whether the key was depressed (code = 5fH) or whether the key
was released (code = 6fH). The second byte is the key number
depressed (0 = key0, 1 = key1, etc.). In this example, the S3016
constantly polls the display for data by repeatedly executing
sfunc10. If no data is sent from the display, the return value
of sfunc10 is "TIME OUT" (3). If data is sent, the return value
is "DONE" (2), at which time the key number is decoded and the
appropriate action is taken.

 0:B63 = sfunc10(2,B60);         /* poll and receive bytes from display */
 1:if (B63 != 1)                 /* time out or done? */
 2:   {
 3:   B64 = B63;                 /* save return value of sfunc10 */
 4:   if (B63 > 3)               /* error code other than timeout? */
 5:      B59 = B63;              /* save error code */
 6:   }
 7:
 8:if (B63 == 2)                 /* were bytes sent from display? */
 9:   {                          /* (return value = DONE) */
10:   if (B60 == 5fH && B62 != 5fH)    /* key depressed? */
11:      {
12:      if (B61 == 0)           /* key #0 depressed ? */
13:         B90 = 1;             /* yes, display message #1 */
14:      else if (B61 == 1)      /* key #1 depressed? */
15:         B90 = 2;             /* yes, display message #2 */
16:      else if (B61 == 2)      /* key #2 depressed? */
17:         B90 = 3;             /* yes, display message #3 */
18:      else if (B61 == 3)      /* key #3 depressed? */
19:         B90 = 4;             /* yes, display message #4 */
20:      else
21:         ;
22:      }
23:   B62 = B60;                 /* update prev key depressed */
24:   }
25:


B059  (-------)  sfunc10  error   code
B060  (-------)  sfunc10 reveive byte#1
B061  (-------)  sfunc10 reveive byte#2
B062  (-------)    key   depress  prev
B063  (-------)  sfunc10 return   value
B064  (-------)  sfunc10 return   value
B090  (-------)          message number
```

APPENDIX A
S3016 PROGRAMMING EXAMPLES

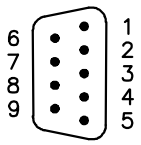
```
S3016E1.LMN

************************************************************************
block:  5 - High-level

This block reads the time and date every five seconds (based on
timer in block 3) and stores the time and date in bytes B160 thru
B165. The time and date is mapped into the bytes as follows:

      B160: hours (1-24)
      B161: minutes (0-59)
      B162: seconds (0-59)
      B163: month (1-12)
      B164: day of month (1-31)
      B165: year (0-99)

Note: A 24 hour format is used for the time and only the
last two digits of the year are provided as well.

In addition, this block also is used to update communications
with the S3012. The S3012 communicates to the S3016 using sfunc12.
The S3016 has a corresponding sfunc12 to receive and transmit bytes
located in the CO-CPU comm file. In this example, the S3012 sends
one word to the S3016 and receives one word from the S3016. The
S3016 has both a dedicated receive buffer (external addresses 1800H
thru 18ffH) and a dedicated transmit buffer (external addressed 1900H
thru 19ffH). when an S3012 transmits data to the S3016, it is stored
in the receive buffer (1800H-18ffH) of the S3016. When the S3012 reads
data from the S3016, it reads it from the transmit buffer (1900H-19ffH)
of the S3016. Thus in this block, the data sent from the S3012 and
stored at 1800H and 1801H is read using sfunc07 and stored at bytes
B76 and B77 (word W76). This block also writes B78 and B79 (word W78)
to the transmit buffer addressed 1900H and 1901H. This is the data
that will be transmitted to the S3012 when the S3012 performs it's
sfunc12.

 0:if (F0 == 1)            /* read time and date every five seconds */
 1:   sfunc02(B160);       /* read time and date and store at B160-B165 */
 2:
 3:/* comm buffer interface */
 4:if (F006 == 1)          /* bus comm O.K.? */
 5:   {
 6:   sfunc07(1800H,B76);  /* read word from S3012 and store at W76 */
 7:   sfunc07(1801H,B77);
 8:   sfunc08(1900H,B78);  /* write word W78 to S3012 */
 9:   sfunc08(1901H,B79);
10:   }
11:


F000  ( (32.0))  display   new   message
F006  ( (32.6))    bus    comm    O.K.
B076  (-------)   byte#1  from    s3012
B077  (-------)   byte#2  from    s3012
B078  (-------)   byte#1   to     s3012
B079  (-------)   byte#2   to     s3012
B160  (-------)   time/   date   buffer
```

```
S3016E1.LCM

************************************************************************
block:  1 - High-level

This is the CO-CPU comm file which is used to respond to an
S3012 sfunc12 call.

Note: The number of words received and the number of words sent
back to the S3012 must match the number of words transmitted from
the S3012 and received by the S3012 in the sfunc12 call located
in the S3012.

The words sent from the S3012 are stored in the receive buffer
of the S3016 (external addresses 1800H-18ffH). The words
transmitted back to the S3012 by the S3016 originate in the
transmit buffer of the S3016 (external addresses 1900H-19ffH).

 0:B70 = sfunc12(1,1);     /* respond to S3012 sfunc12 with and sfunc12 */
 1:if (B70 != 1)           /* valid sfunc12? */
 2:   {
 3:   B72 = B70;           /* save return value */
 4:   if (B70 == 2)        /* was sfunc12 successful (return'"DONE")? */
 5:      F6 = 1;           /* yes, set "bus comm OK" flag */
 6:   else
 7:      F6 = 0;           /* no, clear "bus comm OK" flag */
 8:   }
 9:


F006  ( (32.6))    bus    comm    O.K.
B070  (-------)  sfunc12 return   value
B072  (-------)  sfunc12 return   value
```

APPENDIX A
S3016 PROGRAMMING EXAMPLES

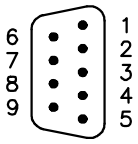
(This Page Intentionally Left Blank)

APPENDIX B
RS-232/RS-422 PIN OUTS/CABLES



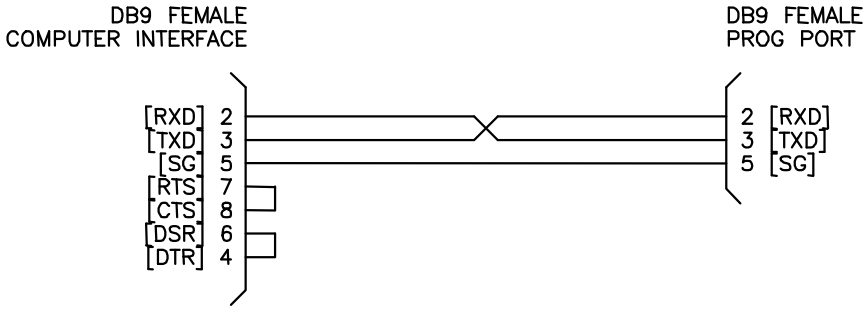
1: NOT USED
2: RXD (RECEIVE DATA – IN)
3: TXD (TRANSMIT DATA –OUT)
4: NOT USED
5: SG (SIGNAL GROUND)
6: NOT USED
7: NOT USED
8: NOT USED
9: NOT USED

PROG Port Pin Out

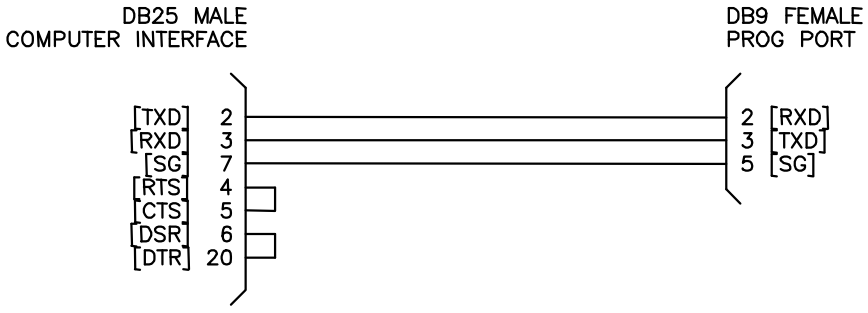


1: NOT USED
2: RS–232 RXD (RECEIVE DATA – IN)
3: RS–232 TXD (TRANSMIT DATA –OUT)
4: NOT USED
5: SG (SIGNAL GROUND)
6: RS–422 TXD+ (TRANSMIT DATA – OUT)
7: RS–422 TXD– (TRANSMIT DATA – OUT)
8: RS–422 RXD+ (RECEIVE DATA – IN)
9: RS–422 RXD– (RECEIVE DATA – IN)

USER Port Pin Out



DB9 (com1) to PROG Port Cable



DB25 (com1) to PROG Port Cable