S3014
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

06/2002

S3014
User's Manual

Copyright © 1990 Systems Engineering Associates, Inc.

All Rights Reserved!

WARNING

To ensure the equipment described by this User Manual, as well as the equipment connected to and used with it, operates satisfactorily and safely, all applicable local and national codes that apply to installing and operating the equipment must be followed. This includes the National Electric Code in the USA and other applicable legislation, regulations, and codes in practice elsewhere. Since codes can vary geographically and can change with time, it is the user's responsibility to determine which standards and codes apply, and to comply with them.

FAILURE TO COMPLY WITH APPLICABLE CODES AND STANDARDS CAN RESULT IN DAMAGE TO EQUIPMENT AND/OR SERIOUS INJURY TO PERSONNEL.

Persons supervising and performing installation or maintenance must be suitably qualified and competent in these duties, and should carefully study this User Manual and any other manuals referred to by it prior to installation and/or operation of the equipment.

The contents of the User Manual are believed to be correct at the time of printing; however, no responsibility is assumed for inaccuracies. In the interests of a commitment to a policy of continuous development and improvement, the manufacturer reserves the right to change the specification of the product or it's performance or the contents of the User Manual without notice.

Copyright © 1990 Systems Engineering Associates, Inc.

All Rights Reserved !

CONTENTS

1. General Description 1

1.1 S3014 Architecture 1
1.2 Program Development 1
1.3 Interface Ports 2
1.4 Fault Detection/Diagnostics 2
1.5 LED Status indications 3

2. Program Structure 5

3. System Configuration 7

4. Variable Types/Memory Map 9

4.1 Variables 9
4.1.1 Flags (F) 9
4.1.2 Bytes (B) 10
4.1.3 Words (W) 11
4.1.4 Port-Pins (P) 11
4.1.5 Constants 12

4.2 Data Memory Map 12
4.2.1 Volatile Data Memory 13
4.2.2 Non-Volatile Data Memory 14

5. Programming Reference 15

5.1 Instruction Set 15
5.1.1 Ladder 15
5.1.2 High-Level (C) 16
5.1.3 Assembly 16

5.2 System Functions 17
5.2.1 System Function Types 18
5.2.2 sfunc07: General External Read 19
5.2.3 sfunc08: General External Write 19
5.2.4 sfunc09: System Fault Routine 20
5.2.5 sfunc13: Serial Network Comm 21
5.2.6 sfunc14: VME Dual-Port RAM Comm 22
5.2.7 sfunc15: VME Dual-Port RAM Byte Read 23
5.2.8 sfunc16: VME Dual-Port RAM Byte Write 23

CONTENTS

6. Extended I/O Operations 25

6.1 Serial Network Communications 25
6.1.1 Communicating on the Network (sfunc13) 25

6.2 VME Dual-Port RAM Interface 27
6.2.1 S3014 Access (sfunc14,15,16) 27
6.2.2 VME Access to Dual-Port RAM 29

7. Fault Detection 31

7.1 Fault Routine Execution 31
7.2 Viewing Fault Codes with SYSdev 31
7.3 Fault Codes 32
7.3.1 Watchdog Timeout (40H) 33
7.3.2 IBM PC to S3014 Comm Fault (42H) 33
7.3.3 Invalid Program Faults (5CH-5DH) 34
7.3.4 User sfunc09 Fault Call (45H) 34
7.3.5 Internal S3014 Fault (43H-5BH) 34

7.4 Serial Network Comm Errors 35
7.4.1 Serial Network Comm Error Codes 35
7.4.2 No Response from Slave (04H-05H) 36
7.4.3 Serial Network Integrity (03H-10H) 36
7.4.4 Address Outside Range Fault (0FH) 36

8. Hardware Confidence Test 37

8.1 Test Performed 37
8.2 Performing the Hardware Confidence Test 37
8.2.1 Equipment Required 38
8.2.2 Executing the Test 38

8.3 Interactive Interface 39

9. Installation 41

9.1 Setting the VME Board Address 41
9.2 Installing S3014 in the VME Rack 42
9.3 Serial Network Installation 43
9.3.1 Wiring the Serial Network 43
9.3.2 Setting the Network Address 45

10. Specifications 47

CONTENTS

APPENDICES

S3014 Programming Example _____ Appendix A
RS-232 Pin-outs/Cables _____ Appendix B

SECTION 1
GENERAL DESCRIPTION

The S3014 VME to S3000 Communications board provides a mechanism for VME bus based systems to communicate with S3000 bus based systems via the S3000 serial network. The S3014 is an A24/D08(EO) slave board that resides in the VME system rack. The board is connected to other S3000 boards via the S3000 serial network.

1.1 S3014 ARCHITECTURE

The S3014 consists of a VME dual-port RAM section, S3000 serial network interface section and a processor section.

The VME dual-port RAM section consists of 2K bytes dual-port RAM which can be read/written from the VME bus and read/written from the processor section of the S3014. VME bus masters transfer data to and from the S3014 simply by reading or writing to this dual-port memory.

The S3000 serial network interface section allows the S3014 to be connected to the S3000-N1 network. This allows up to 32 S3000 boards (S3012s, S3014s, etc.) to communicate with one another.

The processor section executes the user's application program which transfers data to and from the VME dual-port RAM (communicating with the VME bus masters) and transfers data over the S3000 network (communicating to the other S3000 boards). Thus the processor section coordinates the transfer of data between the VME bus masters and the S3000 boards.

1.2 PROGRAM DEVELOPMENT

Programming is implemented using SYSdev51, 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 S3014-1 for program download and online monitoring. The program is developed off-line, compiled, then down loaded into the S3014-1. SYSdev51 allows the S3014-1 to be programmed in a combination of languages: Ladder, High-level (subset of 'C') and Assembly (MCS-51).

SECTION 1
GENERAL DESCRIPTION

1.3 INTERFACE PORTS

The S3014 contains two interface ports: the PROGramming 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 SYSdev51.

SERIAL NETWORK: The S3000-N1 network is a high speed (344KBPS), twisted pair, serial network configured in a master/slave topology. Communications between the S3014 and other S3000 boards (S3012s, S3014s, 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 S3014 or S3012.

1.4 FAULT DETECTION/DIAGNOSTICS

Internal to the S3014 are a series of comprehensive fault detection routines which verify the proper operation of the S3014 at all times. Each detected fault has a corresponding fault code which can be viewed using SYSdev51, 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 S3014. This test is the same test used at the factory to verify proper hardware operation and provides a complete hardware verification of the S3014 board. This test is initiated through SYSdev51.

1.5 LED STATUS INDICATIONS

The following four status LEDs are located on the S3014 faceplate: PWR, RUN, ACC, and COMM. The definitions of these LEDs are as follows:

PWR: "On" when VME power is applied to the S3014.

RUN: "On" steady when the S3014 is running a valid user's 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 S3014 hardware confidence test is executed.

ACC: This LED is flashed every time an access to the VME dual-port RAM by a VME bus master is made. If the LED is on solid, continuous reads/writes are being performed to the VME dual-port RAM. If the LED is "off", no access is being made. This is not a fault LED but simply an indication of activity of the VME dual-port RAM by a VME bus master.

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.

SECTION 1
GENERAL DESCRIPTION

(This Page Intentionally Left Blank)

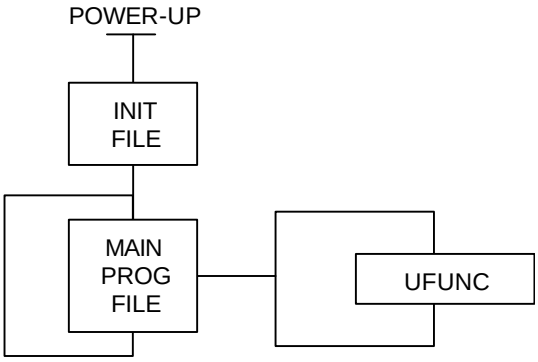
SECTION 2
PROGRAM STRUCTURE

The SYSdev51 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 S3014 user program consists of the following files:

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

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

3) 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 a user function file is called. When this occurs, main program execution is suspended while the user function file is executed. At the completion of the user function, program execution resumes at the point in the main program where the function was called.

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 defines the system or environment that the S3014 program will run in. Unlike other S3000 boards, the S3014 has no optional user defined parameters available for selection. Simply selecting the S3014 as the target board is sufficient to develop the user's application program. See the SYSdev Programming Manual for more details.

SECTION 3
SYSTEM CONFIGURATION

(This Page Intentionally Left Blank)

SECTION 4
VARIABLE TYPES/MEMORY MAP

4.1 VARIABLES

Three classes of variables are used in the S3014. 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).

Four different variable types are available in the S3014: Flags (F), Bytes (B), Words (W), Port-pins (P).

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 S3014 contains 112 flags.

The format of the flag variable is:

Fzzz where: zzz is a three digit flag address (000 to 111).

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, F110, 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 S3014 contains 200 '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 S3014 contains 100 '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 Port-Pins (P):

Port-pins are single bit variables that map directly to specific hardware functions on the S3014 board. These can be input or output hardware functions as defined by the specific port pin (see the following).

The format for port pins is:

Paa where: aa is the two digit port pin (10-17 or 30-37).

Note: The 'P' must be a capital letter and that the port pin address must be two digits.

The S3014 contains only one port-pin that should be accessed by the user program. This is:

P32: watchdog timer reset

Toggling this pin from a "1" to a "0" to a "1" again will reset the hardware watchdog timer, preventing a watchdog timer time-out fault. This is useful when the normal program scan will be suspended by the user program (such as a "for" or "while" loop with a large number of iterations). By toggling P32, a watchdog timer fault can be avoided.

4.1.5 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 '#'.

4.2 DATA MEMORY MAP

The S3014 contains two distinct data memory spaces: 200 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 200 bytes of volatile data memory. The 2K bytes of non-volatile data memory can only be accessed using sfunc07 and sfunc08 (see sections 5.2.2 and 5.2.3).

4.2.1 VOLATILE DATA MEMORY

The memory map for the S3014 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	F104-F111	RESERVED	RESERVED
0046	RESERVED	RESERVED	RESERVED
0047	RESERVED	RESERVED	RESERVED
0048	RESERVED	RESERVED	RESERVED
0049	---------	B049	----
0050	---------	B050	W050
0051	---------	B051	----
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 F111 are mapped into bytes B032 thru B045. 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
1ffeH	---------	----	----
1fffH	---------	----	----

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.

SECTION 5
PROGRAMMING REFERENCE

The following sections provide an overview of the SYSdev instruction set and the system functions available in the S3014. 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 S3014 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.

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 VME bus, communication on the serial network, etc. A summary of the system functions available in the S3014 is as follows:

sfunc07: general external address read
sfunc06: general external address write
sfunc09: system fault routine
sfunc13: serial network communications
sfunc14: VME dual-port RAM block read/write
sfunc15: VME dual-port RAM byte read
sfunc16: VME dual-port RAM byte write

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 S3014: suspended and simultaneous.

Suspended System Function: 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: 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 S3014ally "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 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 1fffH. 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 1fffH) 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.

5.2.3 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 1fffH. 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 1fffH) 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.

5.2.4 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, and User functions.

5.2.5 sfunc13: Serial Network Communications

System function 13 is used to communicate to other S3012s, S3014s or other S3000 nodes on the serial communication network. See section 6.1 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

5.2.6 sfunc14: VME Dual-Port RAM Block Read/Write

System function 14 is used to read and write blocks of data to the VME dual-port RAM of the S3014. See section 6.2 for more details on the use of sfunc14.

General form: sfunc14(#sent,s_srce,s_dest,#rcve,r_srce,r_dest);

Parameters:
- #sent = Number of words transferred to the VME dual-port RAM from the S3014. Variable types: constant (0-255), 'W' or indirect 'W'.
- s_srce = Starting address of stack to be transferred to the VME dual-port RAM. A consecutive number of words (= #sent) is transferred starting with this address. Variable types: 'W' or indirect 'W'.
- s_dest = Starting address in VME dual- port RAM where the words sent are stored. Variable types: constant (0-2046 even address) or indirect 'W'.
- #rcve = Number of consecutive words transferred from the VME dual- port RAM to the S3014. Variable types: constant (0-255), 'W' or indirect 'W'.
- r_srce = Starting address in VME dual-port RAM where words are to be transferred from the VME dual port RAM to the S3014. Variable types: constant (0-2046 even) or indirect 'W'.
- r_dest = Starting address of stack in S3014 where the words from the VME dual-port RAM are stored. Variable types: 'W' or indirect 'W'.

Return value:
- 0 = sfunc14 transfer successful.
- ffH = func14 transfer not successful. Data transferred may not be valid.

Type: suspended

Valid files: Initialization and Main Program only

5.2.7 sfunc15: VME Dual-Port RAM Byte Read

System function 15 is used to read a single byte from the VME dual-port RAM of the S3014. See section 6.2 for more details on the use of sfunc15.

General form: sfunc15(srce,dest);

Parameters:
- srce = The byte address in the VME dual port RAM to be read. Variable types: constant (0-2046) or indirect 'W'.
- dest = Byte address in S3014 where data from VME dual-port RAM is to be stored. Variable types: 'B' or indirect 'B'.

Return values:
- 0 = VME byte read successful.
- ffH = VME byte read not successful, data may be invalid.

Type: suspended

Valid files: Initialization and Main Program only.

5.2.8 sfunc16: VME Dual-Port RAM Byte Write

System function 16 is used to write a single byte to the VME dual-port RAM of the S3014. See section 6.2 for more details on the use of sfunc16.

General form: sfunc16(dest,srce);

Parameters:
- dest = The byte address in the VME dual port RAM to be written to. Variable types: constant (0- 2046) or indirect 'B'.
- srce = Byte address or data in S3014 to be written to VME dual-port RAM. Variable types: constant (0-255) 'B' or indirect 'B'.

Return values:
- 0 = VME byte write successful.
- ffH = VME byte write not successful, data in VME RAM may be invalid.

Type: suspended

Valid files: Initialization and Main Program only.

SECTION 5
PROGRAMMING REFERENCE

(This Page Intentionally Left Blank)

SECTION 6
EXTENDED I/O OPERATIONS

6.1 SERIAL NETWORK COMMUNICATIONS

The serial network provides a means for multiple S3012s/S3014s to communicate with each other. The network described in the following sections is the S3000-N1 network, but other S3000 networks conform to the same general principles.

The network operates in a master/slave topology. One S3012 or S3014 acts as the master and controls all communications on the network. The remaining S3012s/S3014s 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, S3014s or other S3000 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 S3014 from the SYSdev51 Target board Interface menu and is down loaded directly to the S3014 from the IBM PC or compatible running SYSdev51. See section 9.3.2.

6.1.1 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 5.2.5 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 master's) in the entire network has any commands pertaining to network communications.

SECTION 6
EXTENDED I/O OPERATIONS

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 it's 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 it's 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 9.3 for details on installing and wiring the network.

Example:

1) Communicating from the master to a slave:

   Master S3014 main program:

   B050 = 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.

6.2 VME DUAL-PORT RAM INTERFACE

The VME dual-port RAM is a 2K byte dual-port memory which allows VME bus masters to transfer data to and from the S3014. VME bus masters perform these transfers simply by reading or writing data to the dual-port memory. Accesses to the dual-port memory from the S3014 processor are accomplished using the SYSdev system functions sfunc14, sfunc15 and sfunc16.

The S3014 can pass data to the VME master by writing the data to the dual-port RAM (using sfunc14 or sfunc16), the VME processor then reads this data from the dual-port RAM. The VME master passes data to the S3014 simply by writing the data to the dual-port RAM where the S3014 can then read the data (using sfunc14 or sfunc15). Up to 2K bytes can be passed in one transfer cycle. Collisions in the dual-port RAM (access to the same dual-port RAM address by both sides simultaneously) is detected and handled transparently by hardware on the VME side and detected by software on the S3014 side.

6.2.1 S3014 ACCESS TO VME DUAL-PORT RAM (sfunc14,15,and 16)

System functions 14, 15 and 16 are used to interface with the VME dual-port RAM. System function 14 allows up to 255 words to be written and up to 255 words to be read from the dual-port RAM in one command. System function 15 allows one byte to be read from the dual-port RAM and sfunc16 allows one byte to be written to the dual-port RAM. All three system functions suspend program execution when they are executed, thus they are executed just like any other instruction.

The three system functions return with the same return values when executed. If the transfer was successful (no dual-port RAM collision), the return value is zero. If the transfer was not successful (collision with VME side occurred), the return value is ffH. If the return value is ffH, the sfunc should be executed again, the data written or read may not be valid.

System functions 15 and 16 can be used to implement a software handshaking mechanism between the S3014 and VME processor for zero collision data transfers. As an example, using these system functions, the S3014 could "acquire" the dual-port RAM memory by writing a command to a predefined dual-port RAM memory location. This would tell the VME processor that further accesses to the dual-port RAM should be suspended until the S3014 has loaded the dual-port RAM using sfunc14. Once the RAM is loaded, the S3014 would then set the command byte to tell the VME processor to go ahead and read the RAM. This could be implemented for both directions such that the VME could send data to the S3014 using a similar method.

SECTION 6
EXTENDED I/O OPERATIONS

Examples:

1) Reading a byte address in the VME dual-port RAM

   S3014 main program:

   B100 = sfunc15(200,B140);

   Execution:

   The above reads the VME dual-port byte address 200 and stores the data in B140. The return value is stored in B100. The return value should be checked for a read collision (B100 = ffH), and if the collision occurred, the sfunc15 should be performed again, the data in B140 may not be valid.

2) Writing a byte to the VME gateway:

   S3014 main program:

   B110 = sfunc16(1200,B150);

   Execution:

   The above writes the data in B150 to VME gateway address 1200. The return value is stored in B110. The return value should be checked for a write collision (B110 = ffH), and if the collision occurred, the sfunc16 should be performed again, the data stored in gateway address 1200 may not be valid.

3) Transferring a block of data to and from the dual-port RAM

   S3014 main program:

   B060 = sfunc14(10,W100,500,8,800,W130);

   Execution:

   The above writes 10 words (W100 thru W118) to VME dual-port RAM address 500 thru 518. The sfunc then reads 8 VME dual-port RAM addresses (800 thru 814) and stores them at W130 thru W144 in the S3014. The return value is stored in B060 and should be checked for a collision (B060 = ffH). If a collision did occur, the sfunc14 should be executed again, some of the data may be invalid.

6.2.2 VME ACCESS TO DUAL-PORT RAM

The VME side of the dual-port RAM is a simple slave interface accessed as A24/D08(EO) memory by VME masters using the standard non-privileged data address modifier (39H). Accesses to the S3014 occur using the data transfer bus only. The S3014 does not use the interrupt bus or perform any bus arbitration. The memory is read/written to by the VME master using the commands pertinent to the particular VME master board.

The 2K bytes of the S3014 VME interface can be mapped anywhere in the A24 address space. This is done using dip switches mounted on the S3014 (see section 9.1).
The switches select the base address (byte 0 of the 2K dual-port RAM) of the board.

SECTION 6
EXTENDED I/O OPERATIONS

(This Page Intentionally Left Blank)

SECTION 7
FAULT DETECTION

The S3014 contains comprehensive fault detection routines which verify the proper operation of the S3014 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 S3014 is extinguished.
3) Fault code representing the detected fault is saved in internal S3014 memory for viewing with SYSdev51.

7.2 VIEWING FAULT CODES WITH SYSDEV

When a fault occurs, an IBM PC or compatible, running SYSdev51, can be connected to the PROG PORT of the S3014 to view the fault codes. To view the fault codes, perform the following:

1) Connect IBM PC "COM1" port to S3014 "PROG PORT" using the appropriate cable (see appendix B).

2) Initiate SYSdev51 from the DOS prompt and select the user program currently loaded in the S3014.

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 S3014 and displays the following:

Target Board Internal Fault Code
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:

SECTION 7
FAULT DETECTION

Curr Flt: This is the S3014 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.

Last Flt: This is the last S3014 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 S3014 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 S3014.

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 SYSdev51 fault display, detected by the S3014:

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 faults occur when the main program scan time exceeds 40 milliseconds. The cause of this fault range from an unintentional infinite loop entered in the user program to a failure of the S3014 processor.

Trouble-shooting:

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 40 milliseconds to execute (a large number of iterations through the loop). This situation can be corrected by toggling P32, watchdog timer reset, in the loop such that the watchdog timer is reset every iteration of the loop.

3) If the above all verifies, replace the S3014 and try again.

7.3.2 IBM PC TO S3014 COMMUNICATIONS FAILURE (42H)

If an attempt to read the fault codes from the S3014 results in an error code of "42H: Cannot communicate with target board", the PC cannot communicate with the S3014. This is not an internal S3014 fault but instead a fault detected by SYSdev. The cause of this fault ranges from catastrophic failure of the S3014 to a misconnection of the PC to the S3014.

Trouble-shooting:

1) Verify the "PWR" LED on the S3014 is on. If not, verify that power is on the VME rack.

2) Verify that the RS-232 cable is connected to "COM1" on the PC and "PROG PORT" on the S3014.

3) Verify that the RS-232 cable connecting the PC to the S3014 is wired correctly. See appendix B for the pin out of the cable.

4) If the above verifies, replace the S3014 and try again. If the problem still persists, verify the "COM1" port for proper operation (see manual from PC manufacture).

7.3.3 INVALID PROGRAM FAULTS (5CH and 5DH)

The "Program invalid" (5CH) fault occurs when the S3014 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 S3014 is interrupted while program download is in progress.

Trouble-shooting:

1) Dump the user program to the S3014. These faults will clear once the S3014 is loaded with a valid user program.

2) If re-loading the S3014 with the user program does not clear the fault, replace the S3014 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 5.2.4 for details on sfunc09.

7.3.5 INTERNAL S3014 FAULTS (43H,44H,59H-5BH)

The remainder of the fault codes detected by the S3014 represent an internal failure of the S3014. These can range from the RAM battery low to invalid interrupt requests.

Trouble-shooting:

1) Perform the hardware confidence test on the S3014. It may be desirable to remove the suspect S3014 from the system and to install another S3014 to get the application being controlled back up and running. The hardware confidence test can be performed in any VME rack. See section 8 for details on the test.

2) Based on the results of this test, repair S3014, return S3014 for repair, or re-install S3014 in system.

7.4 SERIAL NETWORK COMMUNICATION ERRORS

Unlike the system faults, the serial network communication errors do not cause an S3014 shut-down, 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.

7.4.1 SERIAL NETWORK COMM ERROR CODES

The following is a list of the detected serial network communication 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
9H	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.

Trouble-shooting:

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 9.3.2.

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 S3014.

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.

Trouble-shooting:

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 S3014 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 8
HARDWARE CONFIDENCE TEST

The hardware confidence test allows the S3014 hardware to be verified for proper operation. The test is resident in all S3014s and is initiated through SYSdev51. The hardware confidence test is the same test used at the factory to initially test the production S3014 boards.

Note: The hardware confidence test verifies every aspect of the S3014 hardware with the exception of accesses to the VME dual-port RAM from the VME side. The board should be returned to the factory for further testing if a failure of this section is suspected.

The test is provided to the user to verify whether the S3014 hardware is functional or not. Not as a tool to repair S3014s.

If a fault is detected, the S3014 should be returned to the factory for repair. Any attempt to repair an S3014 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) VME dual-port RAM test
5) Serial network interface test
6) RS-232 PROG PORT test

Each test performs a 100% check of the respective hardware area of the S3014 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 S3014 installed in a users 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 S3014 hardware confidence test, the following is required:

1) IBM PC or compatible with SYSdev51 installed.

2) RS-232 interface cable to connect "COM1" on the PC to "PROG PORT" on the S3014 (see appendix B).

3) Separate VME rack with power supply (+5VDC, +12VDC and -12VDC).

8.2.2 EXECUTING THE TEST

To execute the test, perform the following steps:

1) Install S3014 to be tested in VME rack. Power up rack.

2) Power up PC and enter SYSdev51 from the DOS prompt. Enter any user program name to proceed to the SYSdev51 Main Development Menu.

3) Connect Interface cable to "COM1" on PC and "PROG PORT" on S3014.

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 "S3014 Confidence Test" from the Confidence Test Selection Menu.

7) A prompt verifying continuing will now be displayed.

   Note: Proceeding with this will set the S3014 into test mode, erasing program and data memory. The user application program will have to be downloaded to the S3014 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 answer "y" to the respective prompts.

   Note: Test 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 the Main Test Menu and then into the previous SYSdev menus. The user application program will have to be re-loaded into the S3014, 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 S3014. 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)

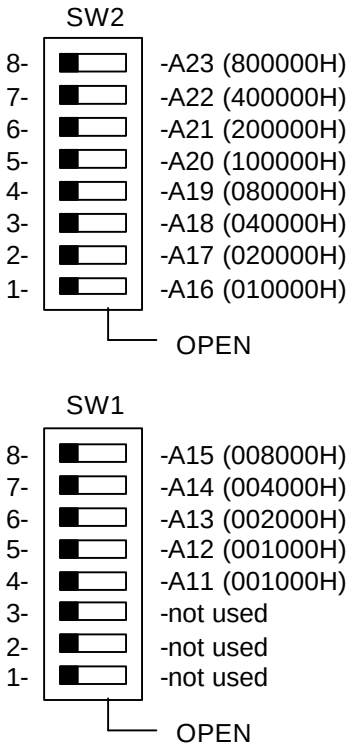
SECTION 9
INSTALLATION

The following sections describe setting the VME board address, installation of the S3014 in the rack, and the installation of the serial network.

CAUTION: The internal components of the S3014 are susceptible to damage by static discharge, just as any electronic components are. When setting the VME address, the S3014 should be handled at an approved antistatic work station. This includes a grounded static dissipative work surface and grounding of personnel handling the board using a grounding wrist strap. When handling the S3014 otherwise, the board should be handled by the faceplate only and preferably in a static shielding bag.

9.1 SETTING THE VME BOARD ADDRESS

The 2K bytes of the VME dual-port RAM can be mapped anywhere in the A24 VME address space. This is done using dip switches SW1 and SW2 mounted on the S3014. These switches select the base address (byte 0 of the 2K dual-port RAM) of the board. The address is selected as shown below:



Note: Depressing the respective switch down on the open side selects the respective bit address, setting it down on the opposite side deselects it.

9.2 INSTALLING S3014 IN THE VME RACK

The S3014 can be installed in any slot in the VME rack that any other slave VME board can be installed in. The bus grant and interrupt acknowledge jumpers should be installed in the slot that the S3014 is installed in. Install the S3014 as follows:

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

2) Set the S3014 VME base address to the desired address. See section 9.1.

3) Install the bus grant and interrupt acknowledge jumpers on the VME back plane at the slot where the S3014 is installed (if the VME system requires these jumpers).

4) Install the S3014 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.

5) Connect the S3014 to the S3000 network by plugging the network field wiring connector into the network comm port, observing the proper keying of the connector.

6) Turn power to the VME rack "on".

7) Set the network address in the S3014. See section 9.3.2.

8) Down-load the user application program to the S3014 using the "Download program to target board" SYSdev menu selection. See the SYSdev programming manual for more details.

To remove the S3014 from the rack, perform the following:

1) Turn power to the VME 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 S3014, power to the rack must be off.

9.3 SERIAL NETWORK INSTALLATION

The serial network installation consists of wiring the network and setting each S3014 on the network with a unique network address. Up to 32 S3012s/S3014s can be installed on one network.

9.3.1 WIRING 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 1MHZ:	.004 db/ft.
Twisted pair, single-shielded	

2) The total wire length of the network cannot exceed 1000 ft.

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 it's 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.

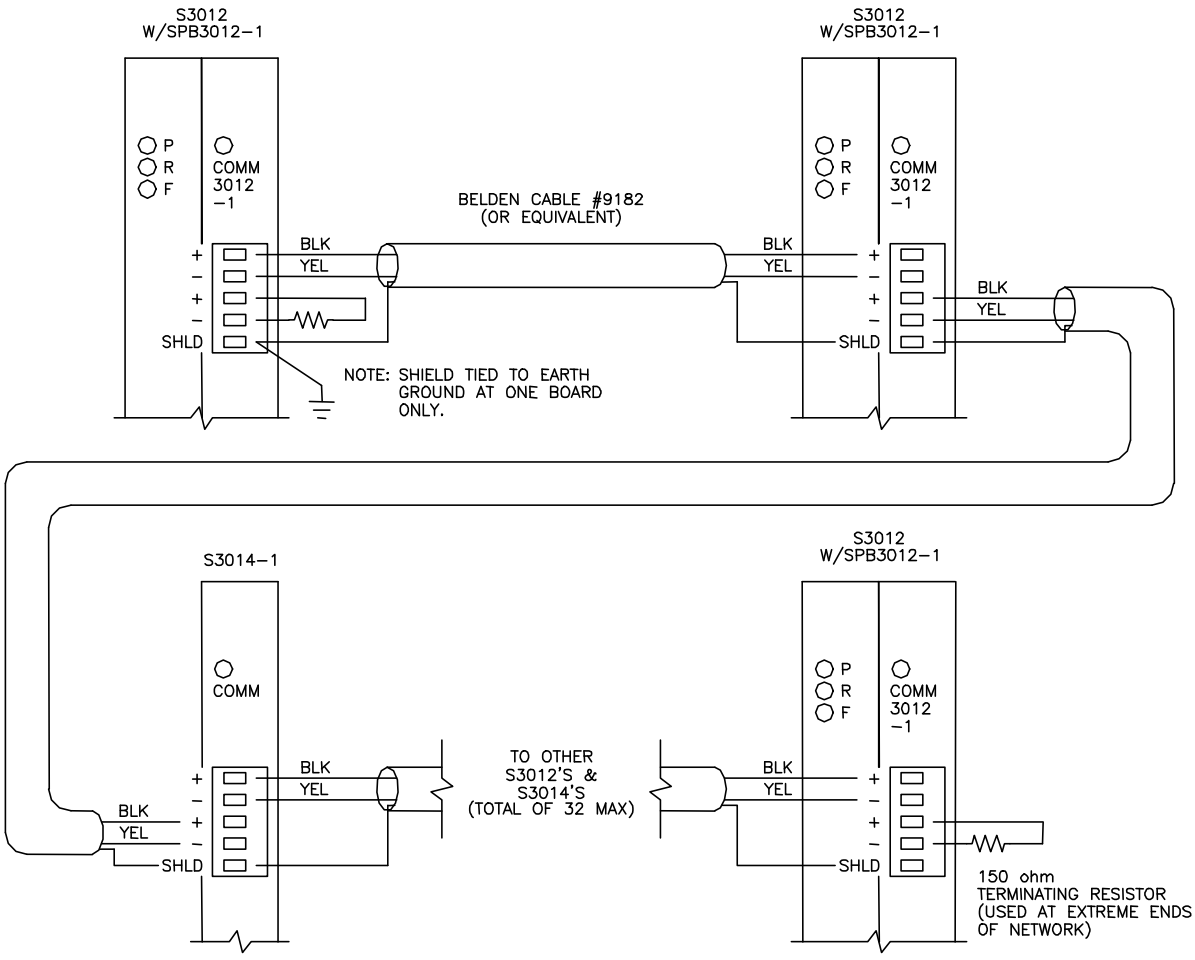


Figure 1 – Typical Network Wiring

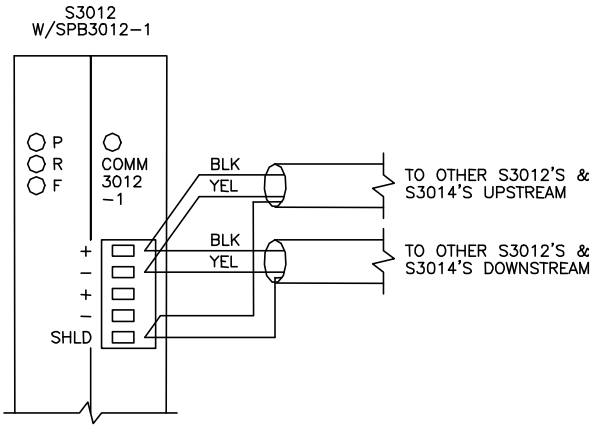


Figure 2

9.3.2 SETTING THE NETWORK ADDRESSES

Each S3012/S3014 on the network must be set with a unique network address between 1 and 32. This is how the S3012s/S3014s can distinguish one node from another. To set the network address for a particular S3014, perform the following:

1) Connect an IBM PC or compatible running SYSdev51 from "COM1" on the PC to "PROG PORT" on the S3014 using the RS-232 interface cable (see appendix B).

2) From the SYSdev Main Development Menu, 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 S3014 and display it in the network display. If the network address is to be changed, follow the directions displayed and enter the new address.

5) Once this is done, power the S3014 down and then power it back up or reload the user application program in the S3014. This resets the S3014, re- initializing it with the new network address.

The above steps must be done for all S3014s on the network. This is true when the network is first installed, and when a new S3014 is added or replaced (that S3014 must have the network address set it in).

SECTION 9
INSTALLATION

(This Page Intentionally Left Blank)

SECTION 10
SPECIFICATIONS

Board Size:

Length:	9.15"
Height:	6.30"
Width:	0.80" (single slot VME)

VME Interface:

VME module type:	slave (memory)
addressing modes:	A24
data transfer type:	D08(EO)
address modifiers:	standard non-privileged data (39H)
memory size/type:	2K bytes/dual-port RAM
memory mapping:	anywhere within A24 address space (selectable via dip switch)
bus connectors:	J1 connector only
VME specification:	Rev C.1

Processor Memory:

Program:	24K bytes battery backed CMOS RAM
Data:	
non-volatile:	2K bytes battery backed CMOS RAM
volatile:	
Flags (F):	112 bits
Bytes (B):	200 bytes
Words (W):	100 words

Interface Ports:

PROG PORT:	
Type:	RS-232
Comm Rate:	9600 BAUD
Serial Network:	
S3000-N1:	
Type:	RS-485
Comm Rate:	344KBPS
# of nodes (max):	32
Isolation:	2000 VRMS
Distance:	1000 ft.
Protocol:	Proprietary

Power Requirements:

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

Temperature Range:

Storage:	0 to 85 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
S3014 PROGRAMMING EXAMPLE

```
VME to S3000 network communications
SYS51 Main Program: S3014E1.LMN


This program is an example of a VME master communicating to an S3012
via the S3014. This program reads the data to be sent to the
S3012 from the VME master via the VME dual-port RAM, transmits this
data to the S3012 over the S3000 network, reads the specified
data from the S3012 over the network and writes this data to the
VME master via the VME dual-port RAM.

S3014 VME dual-port addresses:

   000 = S3014 command: Set to 55 by the VME master to initiate
                        communications with the S3012. Set to 0
                        otherwise.

   002 = S3014 status:  Return value set by S3014. The following values
                        are set by the S3014:

                     1 = BUSY (communications not complete)
                     2 = DONE (communications successful - data is valid)
                  3-16 = communications error code - communication was
                        not successful (see S3014 User's Manual for a
                        description of comm error codes).

   010 - 029 = data stack to be transmitted from VME master to S3012.

   100 - 161 = data stack to be received by the VME master from the S3012.

The communications event (data sent to and received from S3012) is
performed by the VME master as follows:

1) Load data to be sent to S3012 in the transmit stack (010-029).
   See definitions of each address below.

2) Set "S3014 status" (002) to 0. This presets the S3014 status to an
   idle condition (no valid return value).

3) Set "S3014 command" (000) to 55. This initiates the communications.

4) Monitor the "S3014 status" for a return value. Under normal circumstances
   the S3014 will return "1" while the communications is being performed
   and then "2" once it has successfully completed. If an error occurs, the
   S3014 will return an error code (see S3014 status return values above).

5) If the S3014 returned with "2", the data received from the S3012 will
   have been loaded in addresses100-161 and will be valid. If the S3014
   returns wit an error code, the data in 100-161 will not be valid and
   should not be used by the application program.
```

APPENDIX A
S3014 PROGRAMMING EXAMPLE

```
VME to S3000 network communications
SYS51 Main Program: S3014E1.LMN


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

 0:for (B50 = 0, F0 = 0; B50 < 30 && F0 == 0; ++B50)
 1:    {
 2:    if (sfunc15(0,B51) == 0)        /* read S3014 command from VME */
 3:       F0 = 1;
 4:    }
 5:
 6:if (B51 == 0 || F0 == 0)            /* no command from VME? */
 7:   {
 8:   F1 = 0;                          /* reset "VME Transfer Start" */
 9:   F4 = 0;                          /* reset "VME Transfer Complete" */
10:   }
11:
12:if (B51 == 55 && F1 == 0)           /* start transfer command from VME? */
13:   {
14:   B52 = sfunc14(0,W110,100,10,10,W80);
15:   if (B52 == 0)                    /* read data from VME */
16:      F1 = 1;                       /* set "VME Transfer Start" */
17:   B54 = 0;                         /* reset transmit count */
18:   }
19:


F001  ((20.1) )    VME    xfr     start
F004  ((20.4) )    VME    xfr     comp
W110  ((6E)   )   S3014  VMExmit  stack
THRU
W170  ((AA)   )   S3014  VMExmit  stack
```

```
VME to S3000 network communications
SYS51 Main Program: S3014E1.LMN


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

 0:if (F1 == 1 && F4 == 0)       /* communicate with S3012? */
 1:   {
 2:   W60 = 1200;                /* W60 = starting address in S3012 */
 3:                              /* where data is sent to by VME    */
 4:                              /* master. */
 5:
 6:   W62 = 1000;                /* W62 = starting address of data  */
 7:                              /* in S3012 which is read by VME   */
 8:                              /* master. */
 9:
10:   B55 = sfunc13(2,10,W80,*W60,31,*W62,W110);
11:   if (B55 != 1)              /* communicate with S3012. If comm */
12:      F2 = 1;                 /* is done, complete transfer with */
13:   }                          /* VME master. */
14:


F001  ((20.1) )    VME    xfr     start
F002  ((20.2) )  serial   comm    comp
F004  ((20.4) )    VME    xfr     comp
B055  ((37)   )  sfunc13 return   value
W060  ((3C)   )  S3012    xmit   pointer
W062  ((3E)   )  S3012    rcve   pointer
W080  ((50)   )   S3014  VMErcve  stack
THRU
W098  ((62)   )   S3014  VMErcve  stack
W110  ((6E)   )   S3014  VMExmit  stack
THRU
W170  ((AA)   )   S3014  VMExmit  stack
```

APPENDIX A
S3014 PROGRAMMING EXAMPLE

```
VME to S3000 network communications
SYS51 Main Program: S3014E1.LMN


************************************************************************
block:  3 - High-level

 0:if (F1 == 1 && F2 == 1)       /* VME Transfer Start and comm comp? */
 1:   {
 2:   B52 = sfunc14(31,W110,100,0,10,W80);
 3:   if (B52 == 0)              /* transmit data to VME master */
 4:      {                       /* if, transmit was successful, */
 5:      for (B53 = 0, F3 = 0; B53 < 30 && F3 == 0; ++B53)
 6:          {                   /* send transfer done to VME master */
 7:          if (sfunc16(2,B55) == 0)
 8:             F3 = 1;
 9:          }
10:      F4 = 1;                 /* set VME transfer complete */
11:      F2 = 0;                 /* reset comm complete */
12:      }
13:   else
14:      {                       /* if transmit was not successful */
15:      ++B54;                  /* increment attempted transmits */
16:      if (B54 > 10)           /* if more than 10 transmit attempts */
17:         {                    /* have been made, abort transmits */
18:         F4 = 1;              /* set VME transfer complete */
19:         F2 = 0;              /* reset comm complete */
20:         }
21:      }
22:   }
23:


F001  ((20.1) )    VME    xfr     start
F002  ((20.2) )  serial   comm    comp
F003  ((20.3) )  sfunc16  write   valid
W080  ((50)   )   S3014  VMErcve  stack
THRU
W098  ((62)   )   S3014  VMErcve  stack
W110  ((6E)   )   S3014  VMExmit  stack
THRU
W170  ((AA)   )   S3014  VMExmit  stack
```

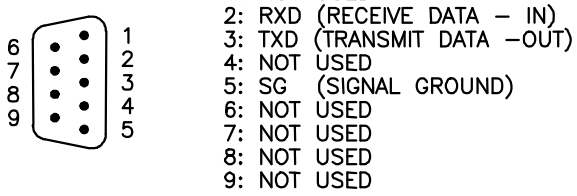
```
VME to S3000 network communications
SYS51 Cross Reference: S3014E1

Typical Cross reference format for variable:

   file:  block-line(usage)

   where: file  = file where variable is referenced.
          block = block number in file.
          line  = line number in block.
          usage = sense that variable is referenced.

   file key: INI = initialization file.
             MNF = main program file.
             TMD = Timed interrupt file.
             COM = co-cpu com interrupt file.
             Uxx = User function file (xx = ufunc num).

   Usage key: * = variable value altered at referenced location.
                  (output coil, timer accumulator, set equal, etc.)

              + = variable used in Assembly block.
                  (usage in Assembly block unknown)

              @ = variable used as pointer in High-
                  level block.

              if *, +, or @ is not associated with variable,
              the variable is not altered at location.
              (contact, timer preset , tested for value, etc.)

   Warning: Cross reference error occurred, cross reference
            is not complete. Compile program to locate error
            in source program.
```

Addr (nicname)	description	location used
F001 ((20.1))	VME xfr start	MNF: 2-0
F002 ((20.2))	serial comm comp	MNF: 2-12*
F003 ((20.3))	sfunc16 write valid	MNF: 3-5* 3-5 3-8*
F004 ((20.4))	VME xfr comp	MNF: 2-0
B055 ((37))	sfunc13 return value	MNF: 2-10* 2-11
W060 ((3C))	S3012 xmit pointer	MNF: 2-2* 2-10@
W062 ((3E))	S3012 rcve pointer	MNF: 2-6* 2-10@
W080 ((50))	S3014 VMErcve stack	MNF: 2-10
THRU		
W098 ((62))	S3014 VMErcve stack	MNF: 2-10
W110 ((6E))	S3014 VMExmit stack	MNF: 2-10
THRU		
W170 ((AA))	S3014 VMExmit stack	MNF: 2-10

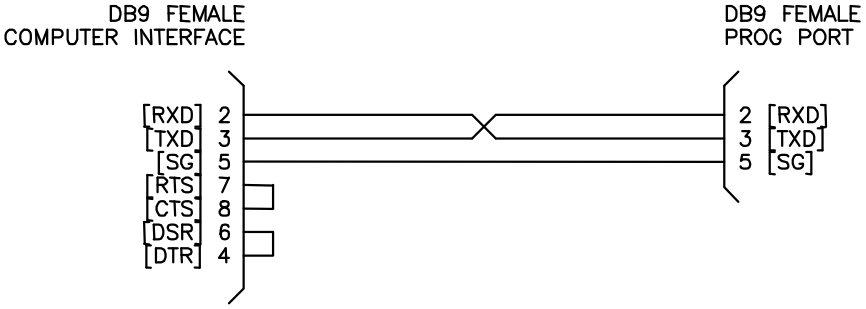
APPENDIX A
S3014 PROGRAMMING EXAMPLE

(This Page Intentionally Left Blank)

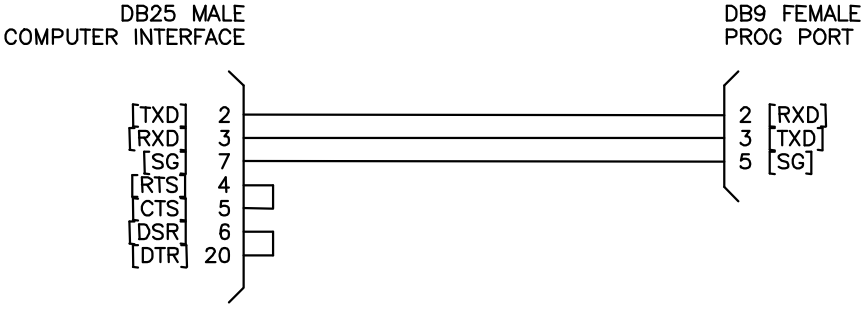
APPENDIX B
RS-232 PIN-OUTS/CABLES



PROG Port Pin Out



DB9 (com1) to PROG Port Cable



DB25 (com1) to PROG Port Cable