M4500
PROGRAM DEVELOPMENT
MANUAL

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

02/2004

M4500
PROGRAM DEVELOPMENT
MANUAL

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.

CONTENTS

CONTENTS

LIST OF FIGURES

SECTION 1
GENERAL OVERVIEW

The M4500 Product Line (generically referred to as the M4500) is a modular design consisting of a series of high performance PLC (Programmable Logic Controller) chassis with a fully integrated PLS (Programmable Limit Switch), digital I/O boards, serial communication board, display/keypads, and power supply. Eight versions are available:

M4500: 4 I/O SLOT PLC/PLS
M4501: 4 I/O SLOT PLC ONLY
M4502: 3 I/O SLOT PLC ONLY WITH DISPLAY/KEYPAD
M4503: 3 I/O SLOT PLC/PLS WITH DISPLAY/KEYPAD
M4508: 8 I/O SLOT PLC/PLS
M4509: 8 I/O SLOT PLC ONLY
M4512: 12 I/O SLOT PLC/PLS
M4513: 12 I/O SLOT PLC ONLY

A typical M4500 system consists of the PLC Processor/Chassis (M4500-M4509), Power Supply (P4500), digital I/O boards (S45XX) as required, Serial Communications boards (S4516) as required, and optional Display/Keypad (D4590 or D4591). The M4502 and M4503 contain a built-in power supply and the D4591 Display/Keypad.

This manual is provided as a reference for programming the PLC section of the M4500 modules. It, in conjunction with the M4500 User's Manual, provides the necessary details to write PLC programs for the M4500. Details on programming the PLS section, as well as installing the M4500, are provided in the M4500 User's Manual.

SECTION 1
GENERAL OVERVIEW

(This Page Intentionally Left Blank)

SECTION 2
PLC FEATURES

The PLC section of the M4500 is a high performance programmable logic controller which incorporates a built-in processor, user program memory, user data memory, RS-232 programming port, interface to the Display/Keypad, and interface to the I/O slots motherboard. The scan time of the PLC section is on the order of 0.25 milliseconds per K with scan times as low as 80 microseconds for short programs. Two additional interrupt inputs allow through-puts even less than 80 microseconds. Program memory consists of 32K bytes of battery-backed CMOS RAM memory. Data memory consists of 8K bytes RAM memory.

The PLC section supports addressing of up to 12 I/O slots.

Note: Geographical addressing is not used. The slot addresses are specified by dip switches on the I/O board themselves. The PLC section is capable of addressing up to 64 bytes at each slot. Memory mapped I/O is incorporated to provide the greatest degree of flexibility in accessing the I/O boards.

Many of the features of the M4500 are accessed via special function variables residing in the data memory space of the M4500. These are byte (B) address that are not used as general data memory but instead enable or activate corresponding features of the M4500. See section 4.3 for complete details.

2.1 PLC PROGRAM STRUCTURE

The PLC section of the M4500 is programmed with SYSdev. The SYSdev programming language is a combination of Ladder, High-level (subset of C) and Assembly (MCS-96). All the files which comprise a SYSdev program are programmed in the same language format. Each file can be written in any combination of the language types. The typical M4500 PLC program consists of the following files:

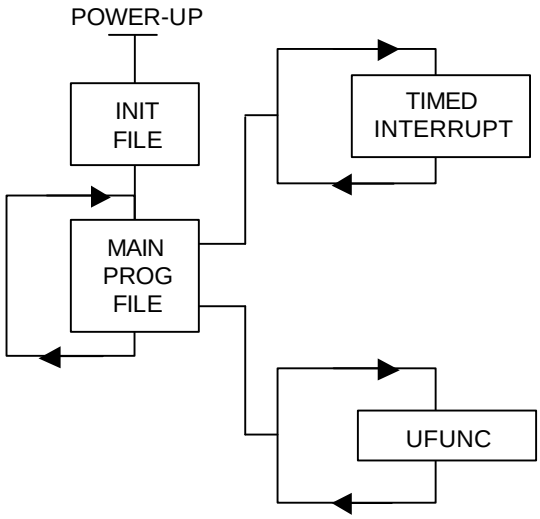
1) Initialization file (optional): executed once at power up.

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

3) Timed Interrupt file (optional): executed once every 0.250 to 65.000 milliseconds as set by the user.

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

5) Input Interrupts (optional): the two input interrupts can be enabled or disabled independently. When the interrupts are enabled, Input0 interrupt calls ufunc00 when activated ("off" to "on" transition of input0) while input1 interrupt calls ufunc01. **Note:** ufunc00 must be created by the user if the input0 interrupt is enabled and ufunc01 if the input1 interrupt is enabled.



Each file is executed sequentially from beginning to end. The main program file is executed (scanned) continuously unless interrupted by the timed interrupt or either of the input interrupts. 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.

Since the M4500 incorporates memory mapped I/O, I/O update is performed by the user's application program. In most applications this is done at the beginning of the main program and/or the beginning of the timed interrupt.

Refer to the SYSdev Program Development Manual for complete details on the SYSdev program language as well as the features of the SYSdev software package.

2.2 INTERRUPT INPUTS

The M4500 modules contain two 10-30VDC differential interrupt inputs. The inputs are 10-30VDC differential inputs which can be enabled as interrupt inputs or disabled and used as standard inputs. When enabled as interrupts, an "off" to "on" transition of the enabled input activates an interrupt call to a user programmed file (ufunc00 for input0 and ufunc01 for input1). This suspends the main program file until the interrupt file execution is completed, at which time program execution resumes at the place in the main file where the interrupt occurs. This mechanism allows ultra fast through-puts to be implemented if required.

The interrupt inputs are enabled or disabled in the system configuration (see sections 3.2 and 3.3).

2.3 ANALOG I/O

Two analog inputs and two analog outputs are built into the M4500. The analog inputs are 0-5 volt which can also be used as 0-20ma and 0-10 volt inputs when external resistors are installed to perform the respective conversion (250ohm for 0-20ma, two resistors as a voltage divider for 0-10VDC). The analog outputs are 0-10 volt which can also be used to drive 0-5 volt inputs using a similar external resistor conversion.

The analog inputs are of 10 bit resolution (0-1023) incorporating high speed conversion (less than 25 microsecond) and are updated once every other main scan. The analog inputs are implemented using memory mapped I/O. The converted analog inputs are mapped to W162 (AIN0) and W166 (AIN1) respectively. The update of the analog inputs is enabled by setting B161.3 to a "1".

The analog outputs are of 8 bit resolution (0-255) and are updated every main scan. Like the analog inputs, the analog outputs are implemented using memory mapped I/O. The digital value to be converted is written to B166 (AOUT0) and B167 (AOUT1) respectively. The corresponding analog output will provide a voltage (0-10V) proportional to the value (0-255) written to these registers.

2.4 DISPLAY/KEYPAD

The D4590 and D4591 displays can be used as a general purpose operator interface or can be used to implement the PLS programming commands. When used as a general purpose interface, complete control of the display is provided through commands accessed through the user program in the PLC section. Commands such as: "position cursor", "clear display", "enter characters into display", "blink character", as well as an ASCII string conversion system function allow easy and complete control of the display directly in the M4500 user's program. The PLS programming commands accessed thru the Display/Keypad are resident in the firmware of the M4500 and can either be enabled or disabled in the user's program. See section 4.5 for complete details.

The keypads are 3 row by 8 column sealed keypads which can either be used as a general purpose operator interface or to implement the PLS programming commands. Key depressed decode is performed automatically by the M4500 with the key number depressed mapped directly to an internal memory location B191 of the M4500 (see section 6.2). The keypad overlay itself simply contains clear windows over the keys. Customization of the keypad is performed by placing a placard behind the overlay (between the overlay and the keys) with the desired key legends. The keys on the D4590 display/keypad are spaced at 0.5" while the keys on the D4591 are spaced at 0.75" spacing.

2.5 SERIAL COMMUNICATIONS BOARD

Serial communications to other equipment is implemented with the S4516 Serial Communications board. The S4516 contains one S3000 serial network interface port and one RS-232/RS-422 User port. Multiple S4516 boards can be installed in one M4500 (up to the number of slots for that particular model) to allow the use of multiple S3000 network ports or RS-232/RS-422 User ports.

SERIAL NETWORK PORT: The serial network port conforms to the S3000-N1 network protocol. This network is a high speed (up to 344KBPS), twisted pair, serial network configured in a master/slave topology. Up to 32 M4500, S3000, or M4000 modules/processors (nodes) can be connected on one network. Communications between the nodes on the network is controlled via commands (sfunc13) in the user application program resident in the node acting as the master (see section 6.3).

USER PORT: This port is available as a general purpose RS-232/RS-422 port accessed under software control of the user program. Access to this port is achieved using sfunc10 and sfunc11 (see section 6.4).

SECTION 3
SYSTEM CONFIGURATION

The system configuration defines the M4500 module configuration that the program will run in. This includes enabling or disabling the input0 and input1 interrupts, and enabling or disabling the timed interrupt. These parameters are all set through SYSdev when the program is developed. See the SYSdev Programming Manual for more details.

3.1 TARGET BOARD

This is used to select the module that the program will be loaded into. For all the M4500 modules, this parameter is always set to M4500. Within the M4500 product family, the SYSdev software package does not distinguish between the different modules (M4503, M4508, etc.). Any differences in features of the various modules is exploited within the user program code (i.e. memory mapped I/O updates, etc.)

3.2 INPUT0 INTERRUPT ENABLE

If the Input0 interrupt is to be used, it must be enabled in the system configuration. The input0 interrupt calls ufunc00 when activated, thus the user must create ufunc00. The ufunc00 file is created and executed just like any other user function file with the exception that it is called when the input0 interrupt input makes an "off" to "on" transition, instead of being called from the main user program. If the input0 interrupt is disabled, interrupt input0 can be used as a standard input (see section 6.5).

3.3 INPUT1 INTERRUPT ENABLE

If the Input1 interrupt is to be used, it must be enabled in the system configuration. The input1 interrupt calls ufunc01 when activated, thus the user must create ufunc01. The ufunc01 file is created and executed just like any other user function file with the exception that it is called when the input1 interrupt input makes an "off" to "on" transition, instead of being called from the main user program. If the input1 interrupt is disabled, interrupt input1 can be used as a standard input (see section 6.5).

3.4 TIMED INTERRUPT

If the timed interrupt file is to be used, it must be enabled in the system configuration. The timed interrupt interval must also be set between 0.250 and 65.000 milliseconds. The timed interrupt file will be called at these intervals, thus the user must create the timed interrupt file. The timed interrupt file is created and executed just as any other file with the exception that it is executed at the specified interval (by interrupting the main program).

Note: The actual timed interrupt execution time must be less than the selected timed interrupt time, otherwise a main program scan watchdog time out will occur.

SECTION 4
VARIABLE TYPES/MEMORY MAP

4.1 VARIABLES

Three classes of variables are used in the M4500. 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 M4500: flags (F), bytes (B), and 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 M4500 module contains 992 flags.

The format of the flag variable is:

Fzzz where: zzz is a three digit flag address (000 to 991).

Note: The leading 'F' must be a capital letter.

Examples: F0, F12, F103, 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 M4500 module contains 6,764 'B' variables.

The format of the byte variable is:

Bzzz where: zzz is the three digit byte address (32 thru 7151).

Note: The leading 'B' must be a capital letter.

Examples: B32, 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 that is addressed at B512 to B7151 are longer than execution times for instructions using flags or byte addresses between B32 and B151. Keep this in mind when using "byte.bit" references.

Examples: B80.0, B100.7, B72.4, 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 M4500 module contains 3,382 'W' variables.

The format of the word variable is:

Wzzz where: zzz is the three digit word address (32 thru 7150).

Note: The leading 'W' must be a capital letter. Also, word addresses are always an even number (divisible by 2).

Examples: W34, W1500, W6050, 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 memory map of the M4500 data memory is shown below:

Address	Valid Variable References			Battery Backed
0032	F000-F007	B032	W032	no
0033	F008-F015	B033	------	no
0034	F016-F023	B034	W034	no
0035	F024-F031	B035	------	no
thru	thru	thru	thru	
0154	F976-F983	B154	W154	no
0155	F984-F991	B155	------	no
0512	---------	B512	W512	yes
0513	---------	B513	------	yes
thru	thru	thru	thru	
7150	---------	B7150	W7150	yes
7151	---------	B7151	------	yes

4.2.1 NON-BATTERY BACKED DATA MEMORY (B32-B155)

The lower 124 bytes of data memory (B32 thru B155) are not battery backed and will not retain data at power down. At power-up or reset, these addresses are cleared.

Note: Flags F0 thru F991 are mapped into bytes B32 thru B155. Bytes B32 thru B155 are also mapped into words W32 thru W154. 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 = B32.0
F001 = B32.1
F002 = B32.2
F003 = B32.3
F004 = B32.4
F005 = B32.5
F006 = B32.6
F007 = B32.7
F008 = B33.0
F009 = B33.1
F010 = B33.2
etc.

The bytes are mapped into the words with even byte addresses 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:

B32 = W32 (low byte)
B33 = W32 (high byte)

4.2.2 BATTERY BACKED DATA MEMORY (B512-B7151)

The upper 6,640 bytes of data memory (B512 thru B7151) are battery backed such that all data residing in this part of memory at power down is retained while power is "off". At power-up these addresses are not cleared, retaining the data that was present before power down. The data in this memory space can also be saved on disk using the data upload menu selection from the SYSdev Target Board Interface menu. The data saved on disk can be downloaded to the M4500 at a later time if necessary. See the SYSdev Program Development Manual for details.

The upper group of memory is referenced as bytes (B) or words (W), no flags are resident in this area of memory. However, any bit within this area can be referenced using the "byte.bit" format outlined in section 4.1.2. The byte (B) variables are mapped into the word (W) variables just as they are in the lower group of memory.

SECTION 4
VARIABLE TYPES/MEMORY MAP

4.3 SPECIAL FUNCTION VARIABLES (SFV's) (Directly addressed)

Many of the features of the M4500 are accessed via certain variables in the M4500. These are referred to as "Special Function Variables" (SFV). Most of the SFV's are fully defined in the related sections that pertain to the function or feature that the respective SFV is used. Section 4.5 defines the System Enable Flags in B161 in detail. The following is a complete list of SFV's of the M4500.

Address	Description
B160	**PLS Command Flags:**
B160.0	Execute PLS command (from PLC)
B160.1	Reserved (Do not access)
thru	
B160.5	Reserved (Do not access)
B160.6	Bypass "Recall" Ack
B160.7	Reserved (Do not access)
B161	**System Enable Flags:**
B161.0	PLS Mode Enable
B161.1	PLS CH0-7 Speed Compensation Enable
B161.2	PLS Keypad Commands Enable
B161.3	Analog I/O Update Enable
B161.4	PLS Update in TIMED(1)/ PLS Update in MAIN(0)
B161.5	PLS Program Commands Enable
B161.6	sfunc13 slave mode enable in S4516
B161.7	sfunc18 interleaved(0)/ executed complete(1)
W162	AIN0 - Analog Input 0 (0-1023)
W164	AIN1 - Analog Input 1 (0-1023)
B166	AOUT0 - Analog Output 0 (0-255)
B167	AOUT1 - Analog Output 1 (0-255)
B170	CH0 PLS Channel byte 0 (CH00-07)
B171	CH1 PLS Channel byte 1 (CH10-17)
B172	CH2 PLS Channel byte 2 (CH20-27)
B173	CH3 PLS Channel byte 3 (CH30-37)
B174	CH4 PLS Channel byte 4 (CH40-47)
B175	CH5 PLS Channel byte 5 (CH50-57)
B176	CH6 PLS Channel byte 6 (CH60-67)
B177	CH7 PLS Channel byte 7 (CH70-77)

Address	Description
W178	Resolver Position (updated in TIMED Interrupt)
W180	Resolver Position (updated in MAIN Program)
W182	Resolver RPM
W184	Resolver Period (msec per revolution)
W186	Resolver Scale Factor
W188	Resolver Offset
W192	Resolver Period (speed compensation)
B191	Depressed Key on Keypad (0=no key, 1-24=key)
B194	Display wait delay (B194=28 for D4590, B194=47 for D4591)
B8158	Keypad Debounce time (in main scans)
W8154	Slave Slot Address for S4516 sfunc13 slave mode
W8156	S4516 Slot Address for sfunc10/11/13
B8159	Current Firmware Revision

PLS Programming (from PLC) SFV's:

B190	Selected PLS Channel Program (0-7)
B212	PLS Command: B212=09: Recall Channel B212=27: Single set-point programming command B212=43: Pulse Train Programming command

For Recall Channel command:

B213	Channel to read (set by user prior to command)
B213	Recall State (0="off", 1="on") (return value)
W214	Position of set-point change of state (return value)

For Single set-point command:

B213	Channel to program set-point
W214	"ON" setpoint (in degrees)
W216	"OFF" setpoint (in degrees)
B218	Program state (0="off", 1="on")

For Pulse Train command:

B213	Channel to program set-point
W214	"START" position (in degrees)
B216	"ON" Duration
B217	"OFF" Duration

General PLS registers:

W246	Current Recall position (used be "Recall" command to indicate last position that change of channel state occurred - location to search from)
B204.0	"Set-point" command active
B204.1	"Pulse Train" command active
B204.2	"Select Channel" command active
B204.3	"Set Config" command Active

Note: The bits in B204 indicate when the respective PLS programming command is active and can be reset unconditionally to defeat the respective PLS command.

4.4 SPECIAL FUNCTION VARIABLES (SFV's) (Indirectly addressed)

Address	Description
2100H	Number of PLS Channels (8,16,32,64)
2120H	Resolver Scale Factor (battery backed)
2122H	Resolver Offset (battery backed)
2124H	Selected PLS Program (battery backed)
2126H	CH0 Speed Compensation (active)
2127H	CH1 Speed Compensation (active)
2128H	CH2 Speed Compensation (active)
2129H	CH3 Speed Compensation (active)
212aH	CH4 Speed Compensation (active)
212bH	CH5 Speed Compensation (active)
212cH	CH6 Speed Compensation (active)
212dH	CH7 Speed Compensation (active)
2146H	Current PLS Channel selected for programming

4.5 SYSTEM ENABLE FLAGS (B161)

The System Enable Flags allow fundamental features of the M4500 to be enabled or disabled dependent on the requirements of the specific application. This allows the overhead scan time to be optimized to the absolute minimum based on the actual features required. The System Enable Flags are located in B161. Individual flags can be set or reset by addressing the individual bits using the "byte.bit" format. The following is complete description of each flag:

B161.0 - PLS Mode Enable: This flag is used to enable or disable the entire PLS section of the M4500. In applications which do not require the PLS (modules without the resolver interface, i.e. M4501, M4502, M4509, M4513), setting this flag to a "0" will disable the PLS section. In this case none of the firmware (code) relating to the PLS will be executed thus reducing the overhead scan time accordingly. If the PLS is to be used, then B161.0 must be set to a "1". The PLS feature will then function normally. At power up, B161.0 is set to a "0" disabling the PLS section. If the PLS is to be used, B161.0 should be set to "1" inside the user's "Init" file.

B161.1 - PLS CH0-7 Speed Compensation Enable: This flag is used to enable the speed compensation of CH0 thru CH07 of the PLS section (see section 3.1.5 of the M4500 User's Manual). If B161.1 is set to "0", speed compensation on these channels will be disabled. If B161.1 is set to "1", the speed compensation will be enabled. At power up, B161.1 is set to a "0" disabling the speed compensation. If speed compensation is to be used, B161.1 should be set to "1" inside the user's "Init" file. B161.1 has no effect if B161.0 is not set to a "1".

B161.2 - PLS Keypad Commands Enable: This flag is used to enable the D4590 and D4591 Display/Keypads as the standard PLS programming keypad (see section 4 of the M4500 User's Manual). If B161.2 is set to "0", the display can be used as a general purpose Display/Keypad with the function of all the keys and display defined by the user program. If B161.2 is set to "1", the Display/Keypad will function as a PLS programmer with the keys defined as outlined in section 4 of the M4500 User's manual and the display automatically updated by the firmware of the M4500 with PLS prompts as necessary. At power up, B161.1 is set to a "0" disabling the Display/Keypad as a PLS programmer. If the Display/Keypad is to be used as a PLS programmer, B161.2 should be set to "1" inside the user's "Init" or "Main Program" file.

B161.3 - Analog I/O Update Enable: This flag is used to enable the conversion of the analog I/O (both the two inputs and two outputs) built into the M4500. If B161.3 is set to "0", the analog I/O conversion will be bypassed, reducing the overhead scan time accordingly. If B161.3 is set to "1", then the analog input conversion will be performed, each input every other scan, with the converted analog input values stored in W162 and W164 for AIN0 and AIN1 respectively. Both analog outputs will be updated on each scan. At power up, B161.3 is set to a "0" disabling the analog input update. If the analog input update is to be performed, B161.3 should be set to "1" inside the user's "Init" file.

B161.4 - PLS Update in TIMED(1)/PLS Update in MAIN(0): This flag determines whether the PLS position and channel update will occur in either the timed interrupt file or main program file. The PLS channel update is equivalent to the input update on standard inputs. The PLS channel update consists of reading the resolver position (with the corresponding scale factor and offset applied) and setting all the PLS channels to the appropriate "0" or "1" state depending on the user programming of these channels at the current position.

If B161.4 is set to a "0", the PLS channel update will be performed at the beginning of each Main Program scan and W180 will be set to the current position. If B161.4 is set to a "1", the PLS channel update will be performed at the beginning of each Timed Interrupt scan and W178 will be set to the current position. At power up, B161.4 is set to a "0", enabling the PLS channel update in the Main Program scan. If the channel update is to be done in the Timed Interrupt, B161.4 should be set to "1" inside the user's "Init" file. B161.4 has no effect if B161.0 is not set to a "1".

B161.5 - PLS Program Commands Enable: This flag is used to enable or disable the programming keys of the D4590 or D4591 Display/Keypads when these are used as the standard PLS programming Keypad (B161.2 set to "1"). The commands disabled when B161.5 is set to a "0" are: "CONF" which includes setting the scale factor, offset, selecting the PLS program, number of timing channels and the speed compensation for CH00 thru CH07, "ENTER SET-POINT", "PULSE TRAIN", and "CLEAR CHAN".

The primary purpose of B161.5 is to allow only authorized personnel to alter the parameters of the PLS section by use of a keyed switch which is input to standard input of the M4500 and mapped to B161.5. When the switch is in the "disable" position, B161.5 would be set to a "0", defeating the use of the command keys. When the switch is in the "enable" position, B161.5 would be set to a "1", and the command keys would function as normal. At power up, B161.5 is set to a "0" disabling the PLS program command keys. B161.5 has no effect if B161.0 and B161.2 are not both set to a "1".

B161.6 - sfunc13 slave mode enable in S4516: This flag enables the slave mode response to communications from other M4500 or S3000 products using the S3000 network (see section 6.3). This flag is used in conjunction with the S4516 communications board installed in the M4500 rack. When B161.6 is set to "0", the network slave response firmware is disabled, reducing the overhead scan time accordingly. The M4500 will not respond to any messages sent from master S3000 or M4500 nodes trying to communicate with the M4500. If B161.6 is set to "1", then the network slave response is enabled and the M4500 will accept communications from other nodes. At power up, B161.6 is set to a "0" disabling the network slave communications. If the M4500 is to act as a slave on the S3000 network, B161.6 should be set to "1" inside the user's "Init" file.

B161.7 - sfunc18 interleaved(0)/executed complete (1): See section 6.1 for complete details on the use of this flag.

4.6 MEMORY MAPPED I/O

'X' and 'Y' variables are not supported in the M4500. Instead the I/O slots of the chassis are mapped into the memory space of the M4500, accessed as Byte(B) variables. The addresses of the different slots are assigned as follows:

Group-Slot	Byte	Address
0-0	0	B7424
	1	B7425
	2	B7426
	3	B7427
0-1	0	B7488
	1	B7489
	2	B7490
	3	B7491
0-2	0	B7552
	1	B7553
	2	B7554
	3	B7555
0-3	0	B7616
	1	B7617
	2	B7618
	3	B7619
1-0	0	B7680
	1	B7681
	2	B7682
	3	B7683
1-1	0	B7744
	1	B7745
	2	B7746
	3	B7747
1-2	0	B7808
	1	B7809
	2	B7810
	3	B7811
1-3	0	B7872
	1	B7873
	2	B7874
	3	B7875

SECTION 4
VARIABLE TYPES/MEMORY MAP

Unlike the S3000 and M4000, the M4500 does not automatically perform the I/O update at the beginning of the program. The I/O update on the M4500 is performed by the user's program. This allows an increased flexibility in how the I/O is updated. To update the I/O, the user should assign some internal Byte(B) variables which will serve as the I/O image (same function as the 'X' and 'Y' variables performed). Input boards are read and stored in the input image (B) bytes while output image (B) bytes are used to store the value of the outputs and then are written to the outputs.

Separate input image bytes should be used for the main program and the timed interrupt if the timed interrupt is used. The same output image bytes can be used in the main program and timed interrupt provided the (B) byte addresses used for the output image are at addresses B32 thru B155. If output image (B) bytes are used in the address ranges B512 thru B7151, then separate output image bytes should be used for the main program and timed interrupt.

Note: The I/O addresses are broken up into two groups of 4 I/O slots each. The I/O address dip switches on the I/O boards consist of two poles which allows the board to be addressed from 0 to 3 (00=0, 01=1, 10=2, and 11=3). The I/O dip switches on the I/O board determine the actual slot the board is addressed as, not the physical slot location in the group the board resides in. Thus within a group, there are 4 unique possible I/O addresses. All the input boards within a group must have unique I/O slot addresses and all the output boards within a group must have unique I/O slot addresses. However an all input board can share the same I/O slot address as an all output board since the input board is read from only and the output board is written to only. This technique allows more than four slots to exist in one group which is the case on the 12 slot M4512 and M4513 chassis.

The following M4500 modules contain the corresponding I/O groups and slots within each group:

Chassis	**I/O Groups**
M4500 (PLC/PLS)	Group0 - 4 I/O slots
M4501 (PLC only)	Group0 - 4 I/O slots
M4502 (PLC only)	Group0 - 3 I/O slots
M4503 (PLC/PLS)	Group0 - 3 I/O slots
M4508 (PLC/PLS)	Group0 - 4 I/O slots Group1 - 4 I/O slots
M4509 (PLC only)	Group0 - 4 I/O slots Group1 - 4 I/O slots
M4512 (PLC/PLS)	Group0 - 4 I/O slots Group1 - 8 I/O slots
M4513 (PLC only)	Group0 - 4 I/O slots Group1 - 8 I/O slots

SECTION 5
PROGRAMMING LANGUAGE REFERENCE

The following sections provide an overview of the SYSdev instruction set and the system functions available in the M4500 module. See the SYSdev Programming Manual for more details on the SYSdev programming language and the operation of the SYSdev software package.

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
   - scan time base
   - 0.01 sec time base
   - 0.10 sec time base
   - 1.00 sec time base

4) Counters

5) Shift Registers

6) Box Instructions
   - Add/Subtract
   - Mult/Divide
   - AND/OR/XOR
   - Shift Left/Right
   - Compare (>,>=,==)
   - Move/Invert
   - ufunc Call
   - Network Comm

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) and words (W). The maximum preset is 255 for bytes (B) and 65535 for words (W).

Valid variables for shift registers are also bytes (B) and words (W). The number of shifts per variable is 7 for bytes (B) and 15 for words (W).

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-96 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 functions such as communication on the serial network, etc. A summary of the system functions available in the M4500 module is as follows:

sfunc03: watch dog timer reset
sfunc04: ASCII string load
sfunc09: system fault routine
sfunc10: USER PORT receive
sfunc11: USER PORT transmit
sfunc13: serial network communications
sfunc18: display write (update)
sfunc19: S4516 configuration

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: 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 basically "back-ground" tasks which are executed while the user application program is executing, with insignificant impact on the user program scan time.

The simultaneous 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 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, Timed Interrupt, and user functions.

5.2.3 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). System function 04 is typically used in conjunction with the display sfunc18 write (update) system function to write ASCII strings to the display.

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'.

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

2) sfunc04(B150,":");

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

3) sfunc04(B1500,"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:

B1500 = 77	(77= ASCII code for 'M')
B1501 = 79	(79= ASCII code for 'O')
B1502 = 84	(84= ASCII code for 'T')
B1503 = 79	(79= ASCII code for 'O')
B1504 = 82	(82= ASCII code for 'R')
B1505 = 32	(32= ASCII code for space)
B1506 = 34	(34= ASCII code for ")
B1507 = 111	(111= ASCII code for 'o')
B1508 = 110	(110= ASCII code for 'n')
B1509 = 34	(34= ASCII code for ")

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 6.1 of the M4500 User's Manual. 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 sfunc10: USER PORT receive

System function 10 receives a consecutive number of bytes from the USER PORT. See section 6.4.1 for a detailed description of the use of sfunc10.

Note: sfunc10 is used in conjunction with the S4516 communications board. See section 5.2.9 for details on configuring the S4516.

General form: sfunc10(#max,dest);

Parameters: #max = This defines the size of the "dest" receive buffer. In essence, this is the maximum number of bytes which can be received between sfunc10 calls. Variable types: constant (1-250), 'B' or indirect 'B'.

dest = The address of the first byte of the sfunc10 receive buffer. The receive buffer is where the bytes received from the USER PORT will be stored. Variable types: 'B' or indirect 'B'.

Return Value: The return value of sfunc10 is the number of bytes which have been received from the USER PORT and stored in the "dest" receive buffer. Unlike sfunc10 in other S3000 boards and M4000 modules, the return value is not BUSY, DONE, or an error code. Once sfunc10 is called, the USER port indefinitely waits for data to be sent to it.

Type: simultaneous

Valid Files: Initialization, Main Program, and user functions.

5.2.6 sfunc11: USER PORT transmit

System function 11 transmits a consecutive number of bytes out the USER PORT. See Section 6.4.2 for a detailed description of the use of sfunc11.

Note: sfunc11 is used in conjunction with the S4516 communications board. See section 5.2.9 for details on configuring the S4516.

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)
- 34 = Invalid S4516 slot address (prior to executing sfunc11, W8156 must be loaded with the address the S4516 is in - see section 4.6)

Type: simultaneous

Valid Files: Initialization, Main Program, and user functions.

5.2.7 sfunc13: serial network communications

System function 13 is used to communicate to other S3012s, S3014s, M4000 modules, or other M4500 nodes on the serial communication network. See section 6.3 for details on the use of sfunc13 and a description of the serial network.

Note: sfunc13 is used in conjunction with the S4516 communications board. See section 5.2.9 for details on configuring the S4516.

General form: sfunc13(slave,#sent,m_srce,s_dest,#rcve,s_srce,m_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'.
- m_srce = Address of source 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 destination 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'.
- s_srce = Starting address of source stack in slave where words will be sent from slave to master. Variable type: 'W' or indirect 'W'.
- m_dest = Starting destination 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 6.4.1 of the M4500 User's Manual for serial network communication error code descriptions).
- 22H = Invalid S4516 slot address (prior to executing sfunc13, W8156 must be loaded with the address the S4516 is in - see section 4.6)

Type: simultaneous

Valid files: Initialization, Main Program, and user functions

5.2.8 sfunc18: display write (update)

System function 18 writes a consecutive number of bytes to the display. See Section 6.1 for a detailed description of the use of sfunc18.

General form: sfunc18(#sent,srce);

Parameters:
- #sent = The number of bytes to write to the display. Variable types: constant (1-250), 'B' or indirect 'B'.
- srce = The address where the first byte written is stored. A consecutive number of bytes (= #sent) is written to the display starting with this address. Variable types: 'B' or indirect 'B'.

Return Values:
- 0 = NOT BUSY/READY
- 1 = BUSY
- 2 = DONE (display updated)

Type: simultaneous with B161.7 set to "0", suspended with B161.7 set to "1".

Valid Files: Initialization, Main Program, and user functions.

5.2.9 sfunc19: S4516 configuration

System function 19 is used to set the S4516 configuration (Network address, Network Baud rate, and USER PORT Baud rate). This must be executed prior to executing either sfunc10, sfunc11, or sfunc13 (generally sfunc19 is executed in the "Init" file of the user program).

Note: For sfunc10, sfunc11, sfunc13, and sfunc19, W8156 must be loaded with the address of the S4516 (byte0 of the group and slot that the S4516 resides in - see section 4.6) just prior to the execution of the sfunc (i.e.):

```
W8156 = 7424;              /* S4516 in slot0, byte0 */
sfunc19(...);
```

This informs the sfunc which slot to execute the system function with.

Note: Multiple S4516s can be installed in one chassis. However, a particular sfunc can act on only one S4516 at a time. Once a system function is initiated with an S4516, a return value of DONE or ERROR must be received before the sfunc can be executed on a different S4516.

The format of the sfunc19 is as follows:

sfunc19(sta_addr,net_baud,user_baud);

Parameters:
- sta_addr = Network station address (1-32) as is normally defined for the S3000 network. Variable types: constant (1-32), 'B' or indirect 'B'.
- net_baud = Network baud rate (1=106K, 2=228K, 3=344K). Variable types: constant (1,2,3), 'B' or indirect 'B'.
- user_baud = USER PORT baud rate (0=9600 baud, 1=19.2K baud). Variable types: constant (0 or 1), 'B' or indirect 'B'.

Return Values:
- 1 = BUSY
- 2 = DONE (S4516 successfully configured)
- 3 = INVALID Parameter (either station address, network baud, or USER port baud rate was invalid)
- 4 = TIMEOUT (no response from S4516)
- 32 = Hardware ACK error from S4516
- 34 = Invalid S4516 slot address (prior to executing sfunc19, W8156 must be loaded with the address the S4516 is in - see section 4.6)

SECTION 5
PROGRAMMING LANGUAGE REFERENCE

(This Page Intentionally Left Blank)

6.1 WRITING (UPDATING) THE DISPLAY

The D4590 and D4591 Display/Keypads of the M4500 contain a separate display processor which actually controls and updates the display. The processor of the M4500 transmits data and control codes to the display processor via sfunc18. This system function is similar to sfunc11 in that the number of bytes to be transmitted to the display and the starting address of these bytes are specified in the system function.

The data sent to the display is a combination of ASCII characters which are to be displayed on the display and control codes which actual control the display (such as "clear" display, position cursor, etc.). Section 6.1.2 contains a list of the display control codes and sections 6.1.3 contains a table of the valid ASCII characters which can be displayed on the display. Both the ASCII data and the control codes are sent to the display via sfunc18. The data and control codes can be intermixed in one sfunc18 call as necessary.

System function 18 can be implemented as either a simultaneous or suspended system function based on the setting of the System Enable flag B161.7. When B161.7 is set to a "0", sfunc18 will function as a simultaneous system function such that once initiated, a return value of BUSY (1) or DONE (2) is returned based on whether the display update is complete or still in progress. New data cannot be sent to the display until the last update is complete (return = DONE). In this mode, one byte of data is sent to the display every scan, thus requiring multiple scans to update the display.

If B161.7 is set to a "1", sfunc18 will function as a suspended system function such that program execution is suspended at the sfunc18 until all the data has been sent to the display.

Note: The display can only except the data sent to it at rate no faster then 28 microseconds per byte for the D4590 and 48 microseconds per byte for the D4591. This delay time is set in the display wait delay SFV B194 (B194 should be set to 28 for the D4590 and 48 for the D4591). The amount of time sfunc18 suspends program execution when executed in this mode is equal to the number of bytes sent to the display times the wait delay. System function 18 will always return with a return value of DONE (2) in this mode.

6.1.1 WRITING DATA TO THE DISPLAY (sfunc18)

Using sfunc18, from 1 to 250 consecutive bytes can be written to the display in one command. System function 18 can be executed as either a simultaneous system function (B161.7=0) or a suspended system function (B161.7=1).

If executed as simultaneous function, once initiated, program execution continues without waiting for the sfunc to complete. Subsequent calls of sfunc18 result in a return value of "BUSY" until the sfunc completes (return = "DONE"). In this mode, since sfunc18 is a simultaneous function, the impact on the user application program scan time is negligible when an sfunc18 is executed. However, since only one byte is written to the display per scan, the update of the display will be slower with the characters actually being seen written out across the display. See example #1 below.

If executed as a suspended function, program execution will be suspended until the display update is complete. This will have an impact on the scan time proportional to the number of bytes written to the display. However, the display will now update much quicker and implementation of sophisticated display drivers is usually easier in this mode. See example #2 below.

The general form of sfunc18 is: sfunc18(#send,srce); where "#send" is the number of bytes to write and "srce" is the starting address of the stack of bytes that will be written to the display. Prior to executing the sfunc18 call, the "srce" buffer is formatted, by the user program, with the appropriate control codes and ASCII character codes. The ASCII character codes can be generated using the sfunc04 ASCII string load command. Both examples #1 and #2 below give examples of formatting the "srce" buffer.

Examples:

1)
```
if (F2 == 1)                    /* update display? */
    {
    B100 = 1fH;                 /* position cursor control code */
    B101 = 5bH;                 /* line 2 - location 27 */
    B102 = 0cH;                 /* display "on", cursor "off" command */
    B103 = 30H;                 /* write character "0" */
    B104 = 32H:                 /* write character "2" */
    if (sfunc18(5,B100) == 2)   /* update display */
       F2 = 0;
    }
```

execution:

The above code writes the characters "02" starting at location 27 of line 2 in the display.

Note: F2 would be set to "1" somewhere else in the user program to perform the display update. F2 stays at "1" until the display update is complete. The user program could then monitor F2 to determine when the display is available (F2 = 0) or BUSY (F2 = 1). This is an example of using sfunc18 in the simultaneous mode.

2)
```
B2000 = 1fH;                /* position cursor control code */
B2001 = 00H;                /* line 1 - location 0 */
sfunc04(B2002,"....message to be displayed in line 1....");
sfunc18(42,B2000);          /* write line 1 to display */
B2000 = 1fH;                /* position cursor control code */
B2001 = 40H;                /* line 2 - location 0 */
sfunc04(B2002,"....message to be displayed in line 2....");
sfunc18(42,B2000);          /* write line 2 to display */
```

execution:

The above writes the entire display (both lines) using an sfunc18 command for each line. The cursor is positioned to the first character position of each line prior to writing the ascii message for the respective line.

Note: The sfunc04s were used to convert the characters "message to be displayed"..etc. to the equivalent ACSII codes into the addresses used as the sfunc18 buffer. These sfunc04s should contain 40 characters each (using spaces to fill in unused display positions) if the entire display is to be updated in one sfunc18 call. This is an example of using sfunc18 in the suspended mode.

6.1.2 DISPLAY CONTROL CODES

The following is a list of valid display control codes. Care should be taken not to send undefined control codes to the display as this may cause unpredictable display operation.

CODE - BINARY 76543210	(HEX)	DESCRIPTION
00000001	(01H)	Reset display, clear enter display area, position cursor at first line, first character (execution time 1.64msec)
00000010	(02H)	Reset display, position cursor at first line, first character (execution time 1.64msec)
000001**DS**		Entry Mode: Specify Cursor Direction: D = 0: Decrement D = 1: Increment Display Shift Mode: S = 0: Display Shift "off" S = 1: Display Shift "on"
00001**DCB**		Display ON/OFF: D = 0: Display "off" D = 1: Display "on" Cursor: C = 0: Cursor "off" C = 1: Cursor "on" Blinking of Character at Cursor Position: B = 0: Blinking "off" B = 1: Blinking "on"
0001**SR**00		Display/Cursor Shift: S = 0: Move Cursor S = 1: Shift Display R = 0: Shift Right R = 1: Shift Left
00011111	(1fH)	Set cursor position: two byte command, second byte is cursor position.

6.1.3 VALID DISPLAYED CHARACTERS

The following is a chart showing the valid character codes which can be displayed on the display. The ASCII character codes 20H thru 7dH can be generated using the sfunc04 ASCII string load command. Examples of this are in example #2 of section 6.1.1.

SECTION 6
USING SYSTEM FUNCTIONS

6.2 KEYPAD INTERFACE

The keypad of the D4590 and D4591 Display/Keypads are 3-row by 8-column sealed keypads. Key depressed decode is performed automatically during the I/O update at the beginning of the main program scan. The key number depressed is automatically updated in SFV byte B191. The corresponding number for each key is shown in figure 6.1. If no key is depressed, B191 is set to zero. If a key is depressed, B191 equals the key number as long as the key is depressed. In this way, the user program can monitor B191 for a depressed key, generating leading and trailing edge transitions of key depression with user logic if desired.

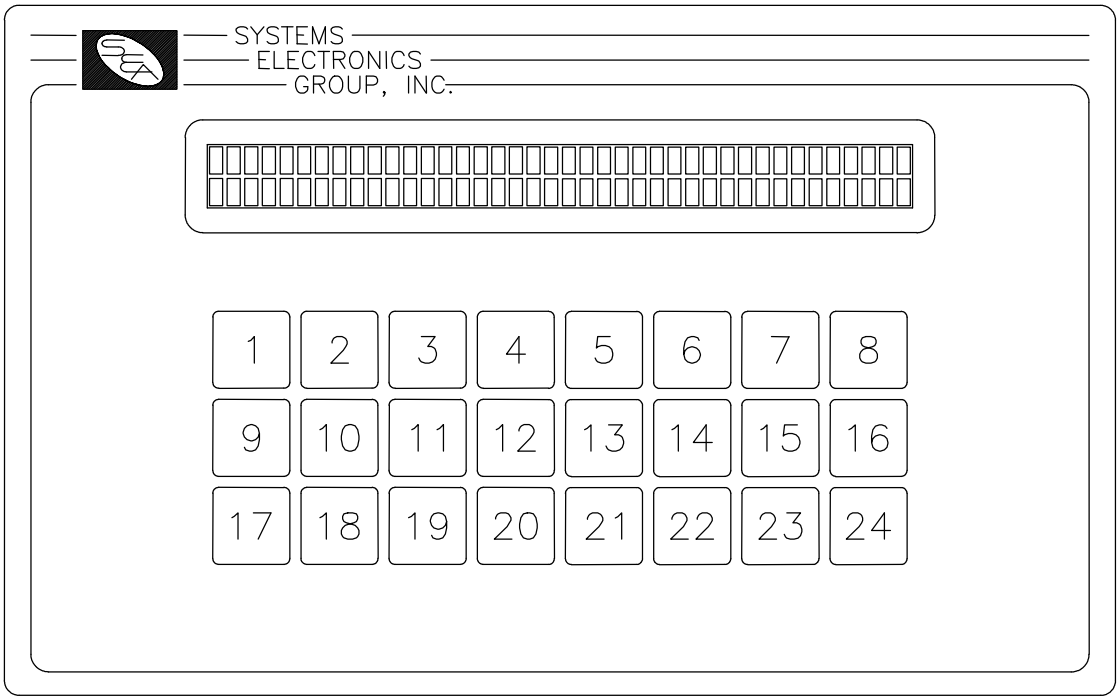


Figure 6.1 - Keypad Key Assignments

6.3 SERIAL NETWORK COMMUNICATIONS (sfunc13)

The serial network provides a means for multiple S3012s, S3014s, M4000 modules or M4500 modules (hereafter referred to as nodes) to communicate with each other. The network operates in a master/slave topology. One module acts as the master node and controls all communications on the network. The remaining nodes 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, M4000 modules, M4500 modules or other S3000 network compatible boards can be installed on one network. These 32 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 of an M4500 module is set via sfunc19 (see section 5.2.9).

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 (m_srce).
4) Starting address of stack, in slave, where the words are to be stored (s_dest).
5) Number of words to be received from slave.
6) Starting address of stack, in slave, where the words will be sent from (s_srce).
7) Starting address of stack, in master, where the words from the slave will be stored (m_dest).

See section 5.2.7 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.

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 6.4.1 of the M4500 User's Manual 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.

In the case of the M4500, the S3000 network is implemented using the optional S4516 communications board. This board provides one S3000 network communications port and one RS-232 USER port. System function 19 must be executed to initialize the S4516 prior to performing any network communications as either a master or a slave. This initializes both the network address and the network baud rate of the M4500 module.

If the M4500 is to be used as a slave on the S3000 network, System Enable flag B161.6 must be set to "1". This activates the network slave firmware of the M4500, allowing it to respond to communication requests from other network nodes.

Examples:

1) Initializing S4516 for communications (S4516 board located in slot02):

```
W8156 = 7552;          /* S4516 located in Slot02 */
sfunc19(5,3,1);        /* S4516 initialized to network node = 5, */
                       /* network baud = 344K, USER port baud */
                       /* rate = 19.2K */
```

2) Communicating from an M4500 to a slave node:

Master M4500 main program:

B070 = sfunc13(4,10,W1000,W1200,5,W1250,W1100);

execution:

The above command transmits the 10 words W1000 thru W1018 in the master M4500 to the slave at network address 4, storing the data in W1200 thru W1218. The slave then transmits 5 words (W1250 thru W1258) to the master, storing this data in W1100 thru W1108. 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 B070. Once the sfunc13 is initiated, the return value of the sfunc13 is "BUSY" (B070 = 1) until the transmission is complete. At that time, the return value is "DONE" (B070 = 2) or an error code (B070 = ERROR CODE) if an error occurred in transmission.

6.4 USER PORT COMMUNICATIONS

The USER PORT is a general purpose RS-232/RS-422 port available on the S4516 communications board for connection to any RS-232/RS-422 user devices. 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. The port is configured as follows: baud rate of 9600 or 19.2K, 1 start bit, 8 data bits, 1 stop bit and no parity.

The USER PORT can be selected as either RS-232 or RS-422 (but not both). This is done by setting a dip switch on the S4516 board (see the S4516 data sheet).

Note: Different pins on the USER PORT connector are used for the RS-232 lines and RS-422 lines (see the S4516 data sheet).

6.4.1 RECEIVING THROUGH THE USER PORT (sfunc10)

Note: sfunc10 functions differently on the M4500 than it does on other S3000 boards or M4000 modules. With other S3000/M4000 boards, sfunc10 must receive the exact number of bytes specified in the sfunc10 call from the user device within a time-out period. In the other S3000/M4000 boards, the return value of sfunc10 specifies whether the sfunc10 is still "BUSY" (has not received the number of bytes specified yet), is "DONE" (all bytes received), or timed-out (not all bytes were sent within the time-out period).

For the M4500, calling sfunc10 essentially enables the USER PORT to receive data from the user device indefinitely. The format of sfunc10 for the M4500 is sfunc10(#max,dest);. Using sfunc10, any number of bytes can be read from a user's ASCII device through the USER port. The return value of the sfunc10 in the M4500 is the number of bytes received since the last sfunc10 call. The bytes that were received are saved in the buffer specified by "dest" in the sfunc10 call.

Note: This buffer should be considered a temporary buffer such that if a string of bytes was expected to be received, the bytes would be "stripped" off this buffer as they were received and saved at some other address locations. The "#max" is the maximum number of bytes that can be saved in the temporary buffer before an overflow occurs. This should be set higher then the maximum number of bytes that can be received between sfunc10 calls.

The operation of sfunc10 in the M4500 allows for a much more flexible format then the previous use of sfunc10 in the other S3000/M4000 boards. With this format, any number of bytes can be sent from the user device at any time. No software handshaking is then required. Also, it is much easier to make the M4500 the slave to the user device then the previous format used.

In the case of the M4500, sfunc19 must be executed to initialize the S4516 prior to performing ufunc10. This initializes the USER PORT baud rate as desired (the default baud rate is 9600 Baud).

See section 5.2.5 for the general form and parameter list of sfunc10.

Examples:

1) Initializing S4516 for communications (S4516 board located in slot02):

```
W8156 = 7552;          /* S4516 located in Slot02 */
sfunc19(5,3,1);        /* S4516 initialized to network node = 5, */
                       /* network baud = 344K, USER port baud */
                       /* rate = 19.2K */
```

2) Receiving through the USER PORT:

B080 = sfunc10(10,B100);

execution:

The above enables the USER PORT to receive data from the user device through the USER PORT. The sfunc10 specifies that the temporary receive buffer is 10 bytes long located at B100 through B109. The return value in B080 is the number of bytes received since the last call to sfunc10 occurred. As an example, assume 3 bytes were received since the last call of sfunc10, these bytes would be loaded into B100, B101, and B102 as they were received. B080 would be set to 3 after the sfunc10 was executed. With B080 = 3, it is now necessary to read B100, B101, and B102 and store them in some other locations since this data will be lost the next time the sfunc10 is called.

6.4.2 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 general form of sfunc11 is: sfunc11(#send,srce); where "#send" is the number of bytes to transmit and "srce" is the starting address of the stack of bytes that will be transmitted. When sfunc11 is initiated, the data in "srce" is loaded into the S4516 communications board and then transmitted out the USER PORT of the S4516.

In the case of the M4500, sfunc19 must be executed to initialize the S4516 prior to performing ufunc10. This initializes the USER PORT baud rate as desired (the default baud rate is 9600 Baud).

Example:

1) B080 = sfunc11(6,B120);

execution:

The above transmits the 6 bytes between B120 and B125 out the USER PORT. The return value of sfunc11 is stored in B080. When sfunc11 is first called, the return value will equal "BUSY" (B080 = 1). Subsequent calls of sfunc11 will result in a "BUSY" (B080 = 1) return value until all 6 bytes have been transmitted, at which time a return value of "DONE" (B080 = 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 the bytes have actually been transmitted. The time it takes for sfunc11 to complete is a function of the number of bytes to be transmitted and the selected Baud rate of the USER PORT.

6.5 USING IN0/IN1 INTERRUPT INPUTS AS STANDARD INPUTS

The INPUT0 and INPUT1 interrupt inputs built into the M4500 can be used as standard inputs instead of interrupt inputs by inserting the following assembly lines into either the main program file or timed interrupt depending on where the inputs are to be updated:

```
        jbs     PORT0,7,in0_1;          map IN0 to F80
        andb    B42,#0feh
        sjmp    in0_2
in0_1:  orb     B42,#01h
in0_2:
        jbs     HSI_STATUS,1,in1_1;     map IN1 to F81
        andb    B42,#0fdh
        sjmp    in1_2
in1_1:  orb     B42,#02h
in1_2:
```

The above assembly block maps IN0 to F80 (B42.0) and IN1 to F81 (B42.1) respectively. IN0 and IN1 can be mapped to any flags desired but note that the flags are accessed as bits within a byte by masking the corresponding bit within that byte to a "0" or a "1". Reminder: Assembly blocks are created by using the "F12: Insert Assembly" selection from the main program editor screen of SYSdev.