M4020
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
www.sea-seg.com

02/2004

M4020
User's Manual

Copyright © 1991 Systems Engineering Associates, Inc.

Revision 2, 1993

All Rights Reserved!

CONTENTS

CONTENTS

CONTENTS

LIST OF FIGURES

APPENDICES

SECTION 1
GENERAL DESCRIPTION

The M4020 is a combination of the M4040 PLS module (PLS section) with the processor and 8 in/8 out digital I/O of the M4010 (PLC section). The PLC section is a high performance programmable logic controller which incorporates a built-in processor, user program (24K bytes) and data memory (2K bytes), 10-30VDC digital inputs, 10-30VDC digital outputs, RS-232 programming port and a serial network interface port.

The PLS section is a high speed programmable limit or cam switch which accepts angular position information in the form of resolver format signals and converts these to digital. The module contains 16 timing channels which can be programmed “on” and “off” at user defined position set-points. The scale factor of the M4020 is programmable from 2 to 512 divisions per revolution while the offset is programmable from 0 to one minus the scale factor. High speed operation is attained with a resolver tracking rate greater than 6,000 RPM and a timing channel output update uncertainty of less than one microsecond to a change in angular position.

1.1 PROGRAMMING

Programming of the PLC section is implemented using SYSdev, an IBM PC or compatible software package which allows the user to create, document, and compile the user application program as well as directly interface to the M4020 for program download and on-line monitoring. The program is developed off-line, compiled, and then downloaded to the module. SYSdev allows the M4020 to be programmed in a combination of languages: Ladder, High-level (subset of C) and Assembly (MCS-51). Typical program scan times are on the order of 0.6 milliseconds per K of user program with scan times as low as 80 microseconds for short programs. Two additional 10-30VDC interrupt inputs allow through-puts even less than 80 microseconds. When using the SYSdev shell, SYSdev is invoked when "M4020 (PLC)" is selected as the target board (see the SYSdev Program Development manual).

SECTION 1
GENERAL DESCRIPTION

Configuration and timing channel programming of the PLS section is performed with PLSdev, a DOS based programming software package which runs on any IBM PC or compatible. PLSdev allows the user to perform the following:

- Configuration of the M4020 including:
  - Scale factor
  - Offset
  - Brake wear compensation enable/parameter programming
  - Motion output enable/parameter programming

- On and Off-line timing channel programming including:
  - arbitrary set-point programming
  - pulse train channel programming
  - timing channel fine tune

- Channel set-points download to M4020

- Channel set-points upload from M4020

- Configuration and Channel set-point print-outs

- M4020 Hardware Confidence test

When using the SYSdev shell, PLSdev is invoked when "M4020 (PLS) is selected as the target board (See SYSdev Program Development Manual). See section 7 for details on the PLS programming commands and PLSdev features.

1.2 DIGITAL INPUTS

The eight digital inputs are 10-30VDC sourcing (true high) which are used to interface to the application inputs such as proximity sensors, push-buttons, etc. The input is “on” (“1”) when the input voltage exceeds 10VDC and is “off” (“0”) when the input voltage is below 5VDC. Individual LED status indication is provided for each input. All inputs are optically isolated and provided with an input filter delay (nominally 1.0 milliseconds).

1.3 INTERRUPT INPUTS

The M4020 contains two interrupt inputs which allow hardware interrupts to be implemented in the user's program. The inputs are 12-30VDC differential inputs which can be enabled as interrupts 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.

1.4 DIGITAL OUTPUTS

The eight digital outputs are 10-30VDC sourcing (true high) which are used to interface to the application outputs such as solenoids, lamps, PLC inputs, etc. Each output is rated at 1 amp DC (continuous) with an inrush (pulsed) current drive capability of 5 amps for 100msec. The sum of the current must not, however, exceed 6 amps. All outputs are optically isolated and contain a transient suppression circuit to protect the output when driving inductive loads. The outputs do not contain output fusing, therefore external fusing should be provided.

1.5 RESOLVER INTERFACE

The M4020 can be used with virtually any type of resolver which incorporates a rotor reference signal and two stator feedback signals. These include resolvers manufactured by C&A, Autotech, Gemco, etc. A dip switch accessible through an access hole on the lower left side of the module selects the desired resolver reference voltage, either 1.45VRMS or 3.70VRMS. For Autotech and Gemco resolvers, the reference should be set at 1.45VRMS, for C&A at 3.70VRMS. For other resolvers, refer to the manufactures specifications to determine which reference should be selected. See section 12.8.1 for details on wiring the resolver to the M4020 and selection of the resolver reference voltage.

1.6 TIMING CHANNELS

The M4020 contains 16 timing channels. Each timing channel can be programmed with up to 50 arbitrary “on-off” set-points or with a pulse train of fixed “on” and “off” divisions throughout the entire channel. All timing channels are mapped to 10-30VDC sourcing (100milliamp) outputs which can be wired to PLC inputs for machine control timing. The status of these channels can also be read internally by the PLC section. The status of each timing channel output is indicated with an individual LED. The timing channel outputs contain short circuit protection such that the detection of a short circuit disables the outputs, with a fault code displayed on the RPM/POSITION display. Once the short circuit condition is removed, the fault code can be reset by toggling the RPM/POSITION toggle switch on the front of the M4020. The outputs will then resume normal operation.

1.7 INTERFACE PORTS

The M4020 contains three interface ports: the serial network comm port, the PROGramming port, and the CHAN port.

SERIAL NETWORK: The serial network port conforms to the S3000-N1 network. This network is a high speed (up to 344KBPS), twisted pair, serial network configured in a master/slave topology. Up to 32 M4000 modules and/or S3000 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.

PROG PORT: The PROG port is an RS-232 port dedicated for on-line monitoring and program download of the PLC section when the M4020 is connected to an IBM PC or compatible running SYSdev.

CHAN PORT: The CHAN port is an RS-232 port dedicated to configuration and channel programming of the PLS section using PLSdev.

1.8 RPM/POSITION DISPLAY/OUTPUT CONNECTOR

A 4-digit RPM/POSITION display is built into the M4020. In addition, an RPM/POSITION output connector is also built in for remote displays or for interfacing to PLCs. If no remote display is used, a switch located on the front of the M4020 is used to select either RPM or POSITION. If a remote display is used, an input on the RPM/POSITION output connector is used to select either RPM or POSITION. See section 9 for complete details on the use of the RPM/POSITION connector.

1.9 BRAKE WEAR COMPENSATION

The brake wear compensation algorithm can be used in conjunction with presses which incorporate a top dead center (TDC) or back dead center (BDC) stop feature. With these presses, a timing signal is used to declutch the press for TDC or BDC stops. However, as the brake wears, the press will no longer stop at TDC but will instead overshoot. When enabled, the brake wear algorithm of the M4020 will automatically adjust the TDC timing signal such that the press always stops at the desired stopping location regardless of brake wear.

The brake wear algorithm is implemented in CH00 and can be enabled or disabled by the user. When disabled, CH00 functions as a standard timing channel. When enabled, the brake wear compensation algorithm is automatically performed for TDC stops. See section 8 for complete details on this feature.

1.10 MOTION (SPEED) OUTPUT

The motion output is a user enabled feature which uses the CH17 output. Two user programmable parameters are used in conjunction with the motion output: low speed threshold and high speed threshold. When enabled as a motion output, CH17 is “on” when the speed (RPM) is between the low and high speed thresholds. When the speed is below the low speed threshold or above the high speed threshold, CH17 is “off”. When disabled, CH17 functions as a standard timing channel.

1.11 DIAGNOSTICS/FAULT DETECTION

The M4020 contains comprehensive fault detection routines which verify the proper operation of the module at all times. Each detected fault has a corresponding fault code which can be viewed using SYSdev, providing a description of the fault and recommended corrective action. The M4020 contains a fault interlock (24VDC, 100mAMP, sinking) output which can be interlocked to the control system for system shut down or annunciation when a fault is detected. In addition to the fault code detection, a hardware confidence test is resident in the module to provide a complete test of the module hardware. This test is initiated through SYSdev and can be used to verify the M4020 for proper operation.

1.12 LED STATUS INDICATIONS

The following four status LEDs are located on the front of the M4020: PWR, RUN, COMM, and FLT. The definitions of these LEDs are as follows:

PWR: "On" when +24VDC power is applied to the M4020.

RUN: "On" steady when the M4020 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 hardware confidence test is executed.

COMM: This LED is flashed every time an access to the serial network is made by any board or module 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 serial network.

FLT: "On" when an internally detected fault has occurred in the M4020. See section 10 for more details on the fault routines and error codes.

SECTION 2
PLC PROGRAM STRUCTURE

The PLC section of the M4020 is programmed with SYSdev. The SYSdev programming language is a combination of Ladder, High-level (subset of C) and Assembly (MCS-51). All the files shown in the following are programmed in the same language format. Each file can be written in any combination of the language types. The typical M4020 user program consists of the following files:

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

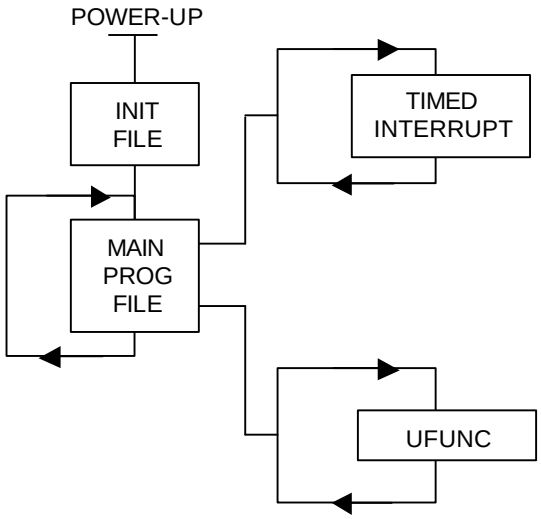
2) Main Program file (required): scanned continuously.

3) Timed Interrupt file (optional): executed once every 0.5, 1.0, or 10.0 milliseconds as set by the user.

4) User Function files (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. 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 an input interrupt is activated. 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.

All M4020 I/O is updated (inputs read, outputs written) at the beginning of each main program scan. These updates are stored in the 'X' and 'Y' I/O image bytes of the module (see section 4.1).

When the timed interrupt is enabled, the 'X' input variables are updated at the beginning of the main program as normal, however, the 'Y' output variables are updated at the beginning of the timed interrupt execution instead of the beginning of the main scan. In addition to these I/O updates, the inputs are read at the beginning of the timed interrupt and stored at special function variables B62-B64 (see Section 4.4.2). This in effect constitutes an immediate I/O for the timed interrupt.

SECTION 2
PLC PROGRAM STRUCTURE

Note: 'Y' output variables cannot be used as coils in the main program if the timed interrupt is enabled. Any outputs that are to be activated by the main program file must be passed to the timed interrupt file as a flag ('F' variable) and then mapped to the 'Y' output in the timed interrupt.

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

SECTION 3
PLC SYSTEM CONFIGURATION

The system configuration defines the configuration of the PLC section of the M4020. This includes defining the serial network baud rate, enabling or disabling the input0 and input1 interrupts, and enabling or disabling the fixed scan mode. 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. This should be set to the M4020. Selecting this enables the complier to generate the appropriate I/O reads and writes corresponding to the available I/O of the M4020.

3.2 NETWORK BAUD RATE

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

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

3.3 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 by reference P32 (see section 4.1.4).

3.4 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 by reference P33 (see section 4.1.4).

3.5 FIXED SCAN TIME MODE

When enabled, the fixed scan time mode allows the user to set the main program scan to a fixed time, either 0.5 milliseconds, 1.0 milliseconds, or 10.0 milliseconds. This allows the main program scan to be used as a high speed time base for either fixed rate sampling or high speed timer time bases (when “scan” time base timers are used).

Note: The actual main program execution time must be less than the selected fixed time, otherwise the scan time will equal the actual scan time rather than the fixed scan time.

If the fixed scan time mode is disabled, the scan time will be a function of the length of the user program and vary as a function of the true/false state of the logic. The fixed scan mode is enabled by selecting 'Y' from the "Enable Fixed Scan or Timed Interrupt?" prompt, then selecting "0 = Fixed Main Scan" from the following prompt.

Note: Both the fixed scan mode and timed interrupt cannot be enabled at the same time.

3.6 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 selected as 0.5, 1.0, or 10.0 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). In addition, all 'Y' outputs are updated at the beginning of the timed interrupt as well as the inputs being read and stored at special function variables B62 - B64 (these are used as the immediate inputs for the timed interrupt).

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.

The timed interrupt is enabled by selecting 'Y' from the "Enable Fixed Scan or Timed Interrupt?" prompt then selecting "1 = TIMED INTRPT" from the following prompt.

Note: Both the fixed scan mode and timed interrupt cannot be enabled at the same time.

SECTION 3
PLC SYSTEM CONFIGURATION

(This Page Intentionally Left Blank)

SECTION 4
PLC VARIABLE TYPES/MEMORY MAP

4.1 VARIABLES

Three classes of variables are used in the PLC section of the M4020. 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).

Six different variable types are available in the M4020: flags (F), bytes (B), words (W), port-pins (P), inputs (X), and outputs (Y).

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 M4020 module contains 104 flags.

The format of the flag variable is:

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

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

Examples: F000, F012, 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 M4020 module 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 M4020 module 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 M4020 module. 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. Port pins can only be referenced in high level blocks.

The following port pins on the M4020 module are mapped to the respective hardware functions:

P32: interrupt input0

The state of interrupt input0 is mapped to this port pin. If interrupt input0 is not enabled as an interrupt, it can be used as a standard (non-interrupt) input.

Note: The state of interrupt input0 is true low logic, thus when the input is "on", P32 will be a "0". When input0 is "off", P32 will be a "1".

P33: interrupt input1

Just as with interrupt input0, interrupt input1 is mapped to port pin P33. Input1 functions identically to input0.

4.1.5 Inputs (X):

Input variables are bytes that contain the data read from the M4000 inputs during the main program I/O update. One 'X' byte is allocated for each input byte, thus the M4020 module has three 'X' bytes allocated for it, one byte for timing channel inputs 00 thru 07, one for timing channel inputs 10 thru 17 and one byte for digital inputs 00 thru 07. The input bytes reside in the I/O image table of data memory and can only be accessed using the 'X' variable designation.

The format for the input byte is:

Xaab where: aa is the two digit I/O address (00-02) and b is the byte at the slot (0 or 1).

Note: The 'X' must be a capital letter and that the I/O address must be two digits (add leading zero). Also, 'X' variables can only be referenced for inputs that are actually available in the module. Any reference to input variables that do not correspond to existing inputs will result in a compiler error.

As with byte variables, individual bits within the 'X' variable can be referenced. These bits correspond to the respective I/O point of the input byte. The form of this is:

Xaab.c where: aa is the I/O address, b is the byte at the slot and c is the bit or input point.

Examples: X010, X000, X010.5, X000.7, etc.

4.1.6 Outputs (Y):

Output variables are bytes which contain the data that is written to M4020 outputs at the beginning of the main program I/O update. One 'Y' variable is allocated for each output byte, thus the M4020 module has one 'Y' variable allocated for it, for outputs 10 thru 17.

The format for the 'Y' variable is:

Yaab where: aa is the two digit I/O address (00-02) and b is the byte at the slot (0 or 1).

Note: The 'Y' must be a capital letter and that the I/O address must be two digits (add leading zero). Also, 'Y' variables can only be referenced for outputs that are actually available in the module. Any reference to output variables that do not correspond to existing outputs will result in a compiler error. 'Y' variables can only be assigned (used as coils) in the main program file but can be referenced (used as contacts) in any file.

As with byte variables, individual bits within the 'Y' variable can be referenced. These bits correspond to the respective I/O point on the output board. The form of this is:

Yaab.c where: aa is the I/O address, b is the byte at the slot and c is the bit or output point.

Examples: Y011, Y011.5, Y011.7, etc.

4.1.7 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 M4020 module 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).

SECTION 4
PLC VARIABLE TYPES/MEMORY MAP

4.2.1 VOLATILE DATA MEMORY

The memory map for the M4020 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	B044	RESERVED
0045	RESERVED	RESERVED	RESERVED
0046	RESERVED	RESERVED	RESERVED
thru	thru	thru	thru
0062	RESERVED	RESERVED	RESERVED
0063	RESERVED	RESERVED	RESERVED
0064	————	B064	W064
0065	————	B065	——
0066	————	B066	W066
thru	thru	thru	thru
0230	————	B230	W230
0231	————	B231	——

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

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

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

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

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

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

4.2.2 NON-VOLATILE (BATTERY-BACKED) DATA MEMORY

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

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

Address	Valid Variable References		
1900H	——	———	———
1901H	——	———	———
thru	thru	thru	thru
1feeH	——	———	———
1fefH	——	———	———

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.

4.3 I/O IMAGE ADDRESSING

The I/O of the M4020 module is mapped to the following I/O image bytes:

I/O Image	I/O Function	
X000	CHAN inputs	0-7
X001	CHAN inputs	10-17
X010	I/O- inputs	0-7
Y011	I/O- outputs	10-17

SECTION 4
PLC VARIABLE TYPES/MEMORY MAP

4.4 SPECIAL FUNCTION VARIABLES

The following variables are used as special function variables. These variables should not be used as general purpose variables within the user program, but only for the purposes described below:

4.4.1 F104: ENABLE RS-232 PROG PORT AS USER PORT

When F104 is a "0", the PROG port on the M4020 is used to download the user program, perform online monitoring, and in general to interface with the PC running SYSdev in the normal PROG port mode. When F104 is set to "1", the PROG port now functions as a user port executing the sfunc10 and sfunc11 user port read and write commands (see section 6.2).
In this mode, the port can be used to interface to an ASCII operator interface or any other device that can accept ASCII data sent via serial RS-232.

Note: When F104 is a “1”, the PROG port will not respond to any commands sent from SYSdev (F104 must be “0” in order to download programs or perform on-line monitoring with SYSdev). Thus, it is highly recommended that an ‘X’ input point is used to set F104 to a “0” or “1”. When the PROG port is to be used to download the program or perform on-line monitoring, the ‘X’ input would be turned “off”, setting F104 to a “0” and enabling PROG port mode. When the PROG port is connected to the user ASCII device, the input would be turned “on”, setting F104 to a “1” and enabling the USER port mode.

4.4.2 B62 - B64: TIMED INTERRUPT IMMEDIATE INPUT VARIABLES

When the timed interrupt is enabled, B62 through B64 are used as the input image bytes of the CHAN inputs and I/O inputs. At the beginning of the timed interrupt, the corresponding inputs are read and the data from these inputs is stored at these variables in the same fashion that the 'X' variables are updated at the beginning of the main scan. Thus, bytes B62 through B64 should be used as the input image bytes inside of the timed interrupt file instead of the 'X' variables.

Note: The 'X' variables are still updated at the beginning of the main scan even when the timed interrupt is enabled.

The I/O of the M4020 module is mapped to the B62 - B64 variables when the timed interrupt is enabled as follows:

Input Image	I/O Function	
B62	CHAN inputs	0 - 7
B63	CHAN inputs	10 - 17
B64	I/O inputs	0 - 7

SECTION 4
PLC VARIABLE TYPES/MEMORY MAP

(This Page Intentionally Left Blank)

SECTION 5
PLC PROGRAMMING REFERENCE

The following sections provide an overview of the SYSdev instruction set and the system functions available in the M4020 module. 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 M4020 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. High-level blocks are implemented as a 57 row X 80 column text array.

The High-level language incorporates the following:

1) Operators:

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

2) Statements:

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

5.1.3 ASSEMBLY

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

5.2 SYSTEM FUNCTIONS

System functions provide the user with a means to perform extended functions such as communication on the serial network, etc. A summary of the system functions available in the M4020 module is as follows:

sfunc04: ASCII string load
sfunc07: general external address read
sfunc08: general external address write
sfunc09: system fault routine
sfunc10: USER PORT receive
sfunc11: USER PORT transmit
sfunc13: serial network communications

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

5.2.1 SYSTEM FUNCTION TYPES

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

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

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

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 sfunc04:ASCII string load command

System function 04 is used to convert the characters in an ASCII string to their equivalent ASCII codes and store these codes in consecutive byte addresses in variable memory (Bxxx variables) or external non-volatile memory (addresses 1900H-1fefH). System function 04 is typically used in conjunction with the USER PORT sfunc11 transmit system function to send ASCII strings to operator interfaces, etc.

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

Parameters: dest = The address where the first ASCII character of the string will be stored. The remaining ASCII characters will be stored in consecutive byte addresses following the first byte address. Variable types: 'B' or constant 1900H-1fefH.

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

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

Return Value: none

Type: suspended

Valid Files: Initialization, Main Program, and user functions

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

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

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

2) sfunc04(B150,":");

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

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

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

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

5.2.3 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 1900H thru 1fefH. This system function reads one byte from the address specified.

General form: sfunc07(ext address,dest);

Parameters: ext address = The 16 bit external RAM address (1900H thru 1fefH) to be read. Variable types: ‘W’ or constant (1900H thru 1fefH).

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.

5.2.4 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 1900H thru 1fefH. This system function writes one byte to the address specified.

General form: sfunc08(ext address,srce);

Parameters: ext address = The 16 bit external RAM address (1900H thru 1fefH) to be written to. Valid variables: ‘W’ or constant (1900H thru 1fefH).

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(W100,B105);

With W100 = 1905H, the above writes the data in B105 to non-volatile data byte address 1905H.

5.2.5 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 10.1. 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.

5.2.6 sfunc10: USER PORT receive

System function 10 receives a consecutive number of bytes from the USER PORT. (PROG port used as USER PORT - F104 set to "1"). See Section 6.2.1 for a detailed description of the use of sfunc10.

General form: sfunc10(#rcve,dest);

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

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

Return Values: 0 = NOT BUSY/READY
1 = BUSY
2 = DONE (receive successful)
3 = TIME OUT (bytes not received)

Type: simultaneous

Valid Files: Initialization and Main Program only

5.2.7 sfunc11: USER PORT transmit

System function 11 transmits a consecutive number of bytes out the USER PORT. (PROG port used as USER PORT - F104 set to "1"). See Section 6.2.2 for a detailed description of the use of sfunc11.

General form: sfunc11(#sent,srce);

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

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

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

Type: simultaneous

Valid Files: Initialization and Main Program only

5.2.8 sfunc13: serial network communications

System function 13 is used to communicate to other S3012s, S3014s or other M4000 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 10.4.1 for serial network communication error code descriptions).

Type: simultaneous

Valid files: Initialization and Main Program only

SECTION 6
SERIAL NETWORK COMMUNICATIONS

The serial network provides a means for multiple S3012s, S3014s or M4000 modules (hereafter referred to as nodes) to communicate with each other. The network operates in a master/slave topology. One S3012, S3014, or M4000 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 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 is set in the M4000 module from the SYSdev Target board Interface menu and is downloaded directly to the module from the IBM PC or compatible running SYSdev. See section 12.7.2.

6.1 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
SERIAL 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 10.4.1 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 12.7 for details on installing and wiring the network.

Example

1) Communicating from the master to a slave:

Master M4000 main program:

B070 = sfunc13(4,10,W080,W100,5,W090,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 5 words (W090 thru W098) to the master, storing this data at W110 thru W118. 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.

6.2 USER PORT (PROG PORT) COMMUNICATIONS

The PROG port can be used as a USER PORT by setting F104 to a "1". (See Section 4.4.1.) The PROG port will then function in the same manner as other S3000 boards equipped with a separate USER PORT (such as the S3012 and S3016). While F104 is set to a "1", the PROG port will be referred to as the USER PORT. As a USER PORT, the PROG port is a general purpose RS-232 port available for connection to any RS-232 user device. Typical applications include: M4020 connection to operator workstations, connection to IBM PC or compatibles for system data acquisition, etc. Communications through the USER PORT is achieved using sfunc10 (USER PORT read) and sfunc11 (USER PORT write). These sfuncs allow any ASCII codes from 0 to 255 to be read from or written to the port.

The baud rate of the USER PORT is preset at 9600 with 8 data bits, 1 stop bit and no parity.

6.2.1 RECEIVING THROUGH THE USER PORT (sfunc10)

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

The device connected to the USER PORT must send the data to the M4020 within a certain time period once sfunc10 is initiated in order to avoid a return value of "TIME OUT". In most applications, software handshaking will be required between the M4020 and user RS-232 device in order to assure the proper number of bytes is sent at the proper time.

Note: The M4020, as the bytes are received through the USER PORT, they are stored directly into the byte addresses specified in the sfunc10 call, there is not an intermediate buffer. Therefore, the return value of sfunc10 should be monitored to determine when all the bytes have actually been received.

The parameters specified in sfunc10 are: the number of bytes to receive and the starting address of the stack to store the bytes at. See Section 5.2.6 for the general form, parameter list and return values of sfunc10.

Example

1) Receiving through the USER PORT:

   Main program:
   B080 = sfunc10(20,B100);

execution:

The above receives 20 bytes from the USER PORT and stores them in B100 thru B119. The return value of sfunc10 is stored in B080. When the sfunc10 is first called, the return value will equal "BUSY" (B080=1). Subsequent calls of sfunc10 will result in a "BUSY" (B080=1) return value until all 20 bytes have been received, at which time a return value of "DONE" (B080=2) is obtained. If the device connected to the USER PORT does not send any or all of the 20 bytes, a return value of "TIME OUT" (B080=3) is obtained after a certain time period.

6.2.2 TRANSMITTING THROUGH THE USER PORT (sfunc11)

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

The parameters specified in sfunc11 are: the number of bytes to transmit and the starting address of the stack of bytes that will be transmitted. See Section 5.2.7 for the general form, parameter list and return values of sfunc11.

Example

1) Transmitting out the USER PORT:

   Main program:
   B080=sfunc11(30,B120);

execution:

The above transmits the 30 bytes between B120 and B149 out the USER PORT. The return value of sfunc11 is stored in B080. When the 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 30 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 30 bytes have actually been transmitted. The time it takes for sfunc11 to complete is a function of the number of bytes to be transmitted.

SECTION 7
PLS PROGRAMMING

7.1 INTRODUCTION TO PLSdev

PLSdev is a DOS based software package used to configure and program the timing channels of the M4020 PLS section. An RS-232 cable connected to COM1 of the IBM PC or compatible running PLSdev is used to interface with the CHAN port of the M4020 module for on-line programming, program upload, download, etc. No other additional hardware is required.

7.1.1 FEATURES OF PLSdev

PLSdev incorporates the following features:

Offline Channel Set-point Programming: set-points for each channel can be entered with easy to use set-point programming commands and saved on disk for download to the M4020 at a later time. This allows the channel programming to be implemented without having an M4020 present.

Online Channel Set-point Programming: using the same set-point programming commands and menus used with the off-line channel programming, the user can alter channel set-points in the M4020 module directly using an RS-232 cable which connects the CHAN port of the M4020 module to COM1 of the IBM PC or compatible running PLSdev. This allows machine timing to be altered while in operation.

PLS Configuration: the configuration of the PLS section is set using PLSdev. This includes defining: the number of PLS timing channels in the module, scale factor, remote display strobe time, CH00 brake wear compensation enable and parameter setting, CH17 speed window enable and parameter setting.

Download Channels to PLS: this allows channels edited in off-line mode or previously uploaded channels to be downloaded to the M4020 module. This feature allows quick replacement of an M4020 module by eliminating the need to reprogram the channel set-points by hand.

Upload Channels from PLS: uploads channel set-points and configuration parameters from the M4020 module to disk files.

Print-outs: the set-points of all channels as well as the PLS configuration can be printed out to provide hard copy documentation.

PLS Hardware Confidence Test: Allows the execution of hardware tests, embedded in the module, to verify the proper operation of the M4020. This is the same test used by the factory to verify the module at completion of manufacturing.

SECTION 7
PLS PROGRAMMING

PLSdev is provided as a stand alone program or with the SYSdev shell. To install or execute PLSdev from the SYSdev shell, refer to the SYSdev Programming Manual for details. To install or execute PLSdev as stand alone program, refer to the following sections.

7.1.2 SYSTEM REQUIREMENTS

PLSdev will run on any IBM PC or compatible with the following minimum system requirements:

1) DOS 3.1 or greater
2) 512K or more RAM
3) Hard disk (not required but recommended)
4) One 3.5" or 5.25" diskette drive
5) COM1 RS-232 port

7.1.3 INSTALLING PLSdev

PLSdev consists of two executable files: PLSdev.exe and S4040T.exe. To install PLSdev, simply install the PLSdev disk in the A: or B: floppy drive and copy “PLSdev.exe” and "S4040T.exe" to the desired directory on the hard drive.

7.1.4 EXECUTING PLSdev

To execute PLSdev, switch to the directory on the hard drive that PLSdev was installed and type:

>PLSdev program<CR>

where “program” is the path and name of the PLS channel program to be edited. The format of “program” should be:

drive:\directory\name

where: drive = the drive letter of the drive your channel program will be stored on.

directory = the user or working directory where your program will be stored.

name = your user program name without any extension.

Note: For the user program name, two user program files are actually created by PLSdev: one that contains the PLS configuration data for the program, another which contains the channel set-point data for the program. These are the files that are used to edit, download, and upload to and from the M4020 module. Thus when referencing the program through PLSdev, no extension should ever be typed in.

If no program name is entered when PLSdev is invoked, the program name will be prompted for by PLSdev before the program proceeds to the main menu. Enter the program name as outlined above to proceed.

7.2 MENUS

The following sections are a description of the various PLSdev menus. In general, the PgUp, PgDn, Home, End, and cursor left, right, up, and down keys all function as defined.

7.2.1 Main Development Menu

1: Offline Channel Setpoint Programming

This selection is used to edit the channel set-points off-line while not connected to an M4020 module. All changes made to the channel set-points are saved in the channel data file for the selected program. The channel set-points can then be downloaded to an M4020 module using the “Download Channels to PLS” selection. To initiate the off-line programming mode, select “1: Offline Channel Setpoint Programming”. This invokes the channel edit menu and loads the existing channel set-points from the channel data file on disk. See section 7.2.2 for a description of the channel edit menu and set-point programming commands. When editing is complete, press “ESC” to return to the main development menu. The modified channel set-points will then be saved in the channel data file on disk for the selected program.

2: Online Channel Setpoint Programming

This selection is used to edit the channel set-points in an M4020 module directly. To initiate the on-line programming mode, connect the COM1 port on the PC running PLSdev to the “CHAN” port on the M4020 to be programmed. Select “2: Online Channel Setpoint Programming”. The channel edit menu will be invoked and the existing channel set-points in the M4020 will be uploaded and displayed in the menu. See section 7.2.2 for a description of the channel edit menu and set-point programming commands.

Note: Any changes to the channel set-points made are updated immediately to the M4020 module. This allows set-point editing during machine operation if desired. When editing is complete, press "ESC" to return to the main development menu. The modified channel set-points will also be saved in the channel data file on disk for the selected program when exiting the on-line mode.

3: Edit PLS Configuration

This activates the PLS configuration menu (See section 7.3). When PLSdev is initially invoked and the program name entered does not exist, the PLS configuration menu is automatically activated. This selection allows the user to modify the system configuration at any time.

4: Download Channels to PLS

This selection downloads both the PLS configuration and channel data files for the selected program to the M4020 module. To download the data to the M4020, perform the following:

Note: Each channel is cleared prior to downloading the set-points for that channel, thus machine operation should be ceased prior to initiating the download.

1) With both the PC running PLSdev and the M4020 powered up, connect COM1 on the PC to the "CHAN" port on the M4020 using the appropriate RS-232 cable.

2) Select this selection from the main development menu. A prompt will appear to verify whether to continue or not. To abort the download press "ESC", otherwise press any key to start the download.

3) While the download is in progress, the channel number which is currently being downloaded will be displayed. Once all channels are downloaded, a dump complete message will be displayed along with a prompt to return to the main menu. Press any key to return to the menu.

4) If the computer was unable to initiate the download to the M4020, a message stating this will be displayed. Verify the RS-232 cable connections between COM1 on the computer and the "CHAN" port on the M4020. Press any key to return to the main development menu and try the download again.

5: Upload Channels from PLS

This selection uploads the set-points for each channel from the M4020 and saves it in the channel data file of the currently selected program.

Note: The configuration data is not uploaded from the M4020. To upload the data form the M4020, perform the following:

1) With both the PC running PLSdev and the M4020 powered up, connect COM1 on the PC to the "CHAN" port on the M4020 using the appropriate RS-232 cable.

2) Select this selection from the main development menu. A prompt will appear to verify whether to continue or not. To abort the upload press "ESC", otherwise press any key to start the upload.

3) While the upload is in progress, the channel number which is currently being uploaded will be displayed. Once all channels are uploaded, an upload complete message will be displayed along with a prompt to return to the main menu. Press any key to return to the menu.

4) If the computer was unable to initiate the upload to the M4020, a message stating this will be displayed. Verify the RS-232 cable connections between COM1 on the computer and the "CHAN" port on the M4020. Press any key to return to the main development menu and try the upload again.

6: Print Channels

Both the PLS configuration data and channel set-points data can be printed out through PLSdev. When this selection is made, a printer selection menu is displayed. Select the appropriate printer to be used. Once this is done, the print-outs selection menu is displayed. The selections are:

1) Print PLS Channel Set-points

   This selection prints the set-points for all the channels. For each channel the following is printed:

   CHANNEL: *number* DESCRIPTION: *users documentation*

   PULSE TRAIN: *yes/no* ON: -- OFF: -- START: --

   SET - POINTS:
   ON OFF
   1: ___ - ____
   etc.

   The above is the set-points data for the respective channel as entered through the “channel edit menu”.

2) Print PLS Configuration

   This selection prints the configuration parameters as entered in the PLS configuration menu.

3) Enter PLS Program Title

   This selection allows the user to enter a title for the program. This title is printed at the top of each page of both the PLS Channel Set-points print-out and the PLS Configuration print-out. The title can be up to 60 characters long and can be composed of any printable characters. When entry of the title is complete, press Enter<CR> to save the title. If the title or changes to the title are not to be saved, press “ESC”.

7: PLS Hardware Confidence Test

This selection is used to invoke the hardware confidence test of the M4020. See section 11 for a complete description of the test.

8: Select PLS Program

Select PLS program is used to change to a different existing PLS program or to create a new program once PLSdev has been invoked. When the selection is entered, the main development menu is cleared and the cursor is placed at the Program Name prompt. Enter the desired program name as was done when PLSdev was initially invoked.

9: File Utilities

The PLSdev program allows you to back-up, restore, make a new directory, and to copy the current program to another program name all while inside PLSdev. Selecting File Utilities brings up a sub-menu with the following choices:

1) Back-up Program

   This allows the current program to be backed up on a diskette in drive A:. Install the back-up diskette in drive A: and press any key when ready. This copies all the files associated with the program to the root directory of the A: drive.

   Note: The files will be stored at the root directory of the diskette, not within a sub-directory. This selection provides a convenient way to back-up your program.

2) Restore Program

   This copies the current program name from the root directory of the A: drive to the drive and directory specified with the current program name. Install the diskette with the program on it in the A: drive and press any key when ready. This copies all the files associated with the program name on the A: drive to the path specified with the program name.

   Note: The program files on the diskette in drive A: must be at the root directory. This selection, along with the back-up selection above, provides a convenient way to copy programs from one computer to another.

3) Make new directory

   This provides a way to make a new user program directory while inside PLSdev. Enter the drive and directory name following the MS-DOS conventions of directory names.

4) Copy program to another program name

   This provides a way to copy the current program name to any disk and directory while also allowing the user to copy to a different program name. Enter the drive, directory, and new program name using the MS-DOS conventions for directory and file names. Do not enter an extension with the program name. This copies all the files associated with the program to the different directory and program name. This selection can be used to copy the current name to another drive and directory (when the program name entered is the same as the current program name). This is also used to copy the program to a new program name. For instance when one program is similar to another completed program, simply copy the old program to the new program name and edit as required.

7.2.2 Channel Edit Menu

This menu is invoked for both off-line and on-line programming and provides a mechanism to enter and edit the set-points for the channels. The menu contains both information fields and function key commands. The information fields are defined as follows:

CHANNEL: This is the number (in octal) of the currently selected and displayed channel. When the channel edit menu is initially invoked, channel number 00 is selected and displayed. The “F1: Next Chan”, “F2: Prev Chan”, and “F3: Select Chan” commands are used to select a different channel number.

DESCRIPTION: This field contains the user entered description or channel name which is associated with the channel number. The “F4: Doc Chan” command is used to enter or edit this description.

PROG MODE: This field displays the program mode, either OFFLINE or ONLINE depending on whether the channel edit menu was invoked from the Offline or Online selection of the main development menu.

SET-POINTS: This a 10 row by 5 column, 50 element array where the set-points are entered using the various set-point programming commands.

Note: If the channel is not programmed as a pulse train (PULSE TRAIN = NO), that up to 50 unique set-points can be entered in the channel. If the channel is programmed as a pulse train (PULSE TRAIN = YES), up to the scale factor divided by two number of set-points (ON = 1, OFF = 1) can be programmed in the channel. In this case only the first 50 set-points would be displayed in the channel, however the channel would be programmed through-out with the “on” and “off” duration specified. See section 7.4 for complete details on the set-point programming commands.

If the channel is not programmed as a pulse train, a large block cursor is placed in the "ON" field of the currently selected set-point to be edited. This cursor can be moved to any set-point number using the cursor (arrow)
left, right, up, and down keys. Whichever set-point the block cursor is located at is the set-point that the various set-point programming commands will operate on.

If the channel is programmed as a pulse train, the cursor will not be display at all. The only commands which are valid once a channel is programmed as a pulse train are the "F5: Pulse Train" and "F8: Clear Chan" commands. Thus the cursor is not used to select set-points since the commands that operate on individual set-points are not valid.

If no set-point is programmed at a given set-point number, the field is displayed as "___-___". Otherwise, as an example, the set-point will be displayed as 020-040, where 020 is the location the channel turns "on" and 040 is where the channel turns "off" for the given set-point.

SCALE FACTOR: This is the scale factor as entered in the PLS configuration.

Note: The scale factor cannot be changed from the channel edit menu, but is displayed only for reference.

MESSAGE: This is field which displays various status messages, informing the user of invalid commands (i.e. "invalid set-point"), operations in process (i.e. "loading file.."), etc.

OFFSET: The current resolver offset of the selected program. This is entered using the "F10: Set Offset" command.

PULSE TRAIN: Defines whether the channel is programmed as a pulse train (YES) or not (NO). The channel defaults to "NO" until the "F5: Pulse Train" command is executed. Once programmed as a pulse train, the channel must be cleared using the "F8: Clear Chan" command to reset PULSE TRAIN to "NO". See section 7.4 for details of the pulse train command.

ON: "On" duration, in degrees, of the pulse train.

OFF: "Off" duration, in degrees, of the pulse train.

START: Starting location of the pulse train. The Channel will be programmed through-out with the "on" and "off" durations starting at the "Start" location.

SECTION 7
PLS PROGRAMMING

The function key commands of the channel edit menu are defined as follows:

F1: Next Chan

Selects the next highest numbered channel for editing. When selected, the channel number will increment by one and the set-points and data for that channel will be displayed. The “PgDn” key also performs the same function as the “F1: Next Chan” key.

F2: Prev Chan

Selects the next lowest numbered channel for editing. When selected, the channel number will decrement by one and the set-points and data for that channel will be displayed. The “PgUp” key also performs the same function as the “F2: Prev Chan” key.

F3: Select Chan

Used to select any channel number for editing. When selected, the current channel number will be cleared and the cursor will be placed in the CHANNEL field. Simply type in the desired channel number, in octal, and press Enter<CR> to accept. The set-points and data for that channel will then be displayed.

F4: Doc Chan

Used to enter the channel description or name in the DESCRIPTION field. This is a user definition of the channel and will be associated with the channel at all times (print-outs, etc.). Up to 20 printable characters can be entered in this field. Once the name is entered, press Enter<CR> to accept.

F5: Pulse Train

Used to program the channel as a pulse train. See section 7.4 for details.

F6: Fine Tune

Used to fine tune (increment or decrement) the selected set-point. See section 7.4 for details.

F7: Clear SetPnt

Clears the currently selected set-point (set-point designated with cursor). No other set-points are affected by this command.

F8: Clear Chan

Clears all the set-points in the channel. Also used to clear the channel and reset the pulse train mode to “no” when a channel has been programmed as a pulse train.

F9: Set Offset

Used primarily in the on-line mode to electronically zero the resolver shaft at the machine zero. To set the offset, locate the machine at machine zero. Read the resolver shaft position from the front of the M4020, select “F9: Set Offset” and enter the position into the offset field. The actual offset number, required to make this the zero position, will than be calculated by PLSdev and downloaded to the M4020. The actual offset (which may not equal the entered position) will be displayed in the offset field and the M4020 will then display zero as the position.

7.3 PLS CONFIGURATION

The PLS configuration is used to define the following PLS parameters in the M4020: number of PLS channels, scale factor, remote display strobe time, CH00 brake wear compensation parameters, and CH17 speed window parameters. The parameters are all set through the “Edit PLS Configuration” menu selection.

7.3.1 NUMBER OF PLS CHANNELS

This is the number of PLS timing channels available in the specific module being configured. The number of channels should be set as follows:

for M4020: set number of channels = 16

7.3.2 SCALE FACTOR

The scale factor is the desired number of divisions per revolution. This is programmable from 2 to 512. For 360 degrees per revolution, the scale factor should be set to 360.

7.3.3 REMOTE DISPLAY STROBE TIME

The remote display strobe time is the time that the strobe outputs on the remote display connector are activated to strobe the respective digits for the remote display. The number entered, between 1 and 255, is a preset for a timer with a 2.5msec time base and represents the time the strobe output is "on". See section 9 for more details on the remote display output.

7.3.4 CH00 BRAKE WEAR COMPENSATION

If the brake wear compensation algorithm available in CH00 is to be used, it must be enabled in the PLS configuration. If enabled, the following additional parameters must be set:

DESIRED STOPPING LOCATION: This is the desired stopping location, in degrees, of the press when a normal top dead center or back dead center stop is performed.

ALLOWED STOPPING ERROR: This is the amount of deviation, in degrees, from the desired stopping location, that is allowed before the brake wear compensation algorithm modifies the CH00 timing.

CH00 TIMING SIGNAL WIDTH: This is the width, in degrees, that the compensation algorithm will program the CH00 set-point when it is modified.

COMPENSATION ENABLE WINDOW: This is the window in which the brake wear algorithm will be allowed to move the CH00 set-point for brake wear compensation.

See section 8 for more details on the CH00 brake wear compensation algorithm and the use of the above parameters.

If the CH00 brake wear compensation is disabled, CH00 will function as a standard timing channel and the brake wear parameters above will be ignored.

7.3.5 CH17 SPEED WINDOW

If CH17 is to be used as a speed window, it must be enabled in the PLS configuration. If enabled the following additional parameters must be set:

LOW SPEED THRESHOLD: If the speed is below this threshold, CH17 is “off”.

HIGH SPEED THRESHOLD: If the speed is above this threshold, CH17 is “off”.

The two thresholds above define a window such that if the speed is between or equal to the low and high thresholds, CH17 is “on”. If the speed is below the low or above the high threshold, CH17 is “off”. Both thresholds are defined in RPM.

If the CH17 speed window is disabled, CH17 will function as a standard timing channel.

7.4 CHANNEL SET-POINT PROGRAMMING COMMANDS

Three channel set-point programming commands are available through PLSdev. These are: single set-point programming, set-point fine tune, and the pulse train programming command. The single set-point command allows the user to simply type in the complete set-point, both “on” and “off” parameters, at the currently selected set-point. The fine tune command allows the user to either increment or decrement by one the “on” or “off” set-point parameter of the currently selected set-point. The pulse train programming command allows the user to program the entire channel with a pulse train of a fixed “on” and “off” duration through-out.

7.4.1 SINGLE SET-POINT PROGRAMMING COMMAND

The single set-point command is used to enter unique set-points in a channel when that channel is not programmed as a pulse train (PULSE TRAIN = NO).

Note: When the channel is programmed as a pulse train, the single set-point command cannot be used to modify individual set-points in the channel. To program a single set-point, locate the set-point cursor (large block cursor) at the set-point to be modified (the large cursor will always be located in the "on" parameter of the set-point). Enter the set-point as follows:

1) Type in "on" parameter in degrees (cursor will change to the small data entry cursor)

2) Press Enter<CR> or '-' to enter the "on" parameter.

3) Type in "off" parameter in degrees.

4) Press Enter<CR> to enter the "off" parameter.

5) The cursor will change back to the large set-point cursor and automatically advance to the "on" parameter of the next set-point. The set-point has now been entered.

6) If an invalid set-point number is entered, a message stating this will be displayed in the "message" field of the channel edit menu. Re-enter a proper set-point value or press "ESC" to abort the set-point entry.

Note: The set-point values entered must be less than the scale factor. Set-points are also not allowed to overlap any existing set-points already entered in the channel.

Example #1: The following key sequence enters the set-point 020-040 at set-point number 1:

Key Depressed	Set-point Field
	___ - ___
2	2__ - ___
0	20_ - ___
-	020 - ___
4	020 - 4__
0	020 - 40_
Enter<CR>	020 - 040

Example #2: The following example programs the entire channel "on" by specifying both the "on" and "off" parameters to 000:

Key Depressed	Set-point Field
	___ - ___
0	0__ - ___
Enter<CR>	000 - ___
0	000 - 0__
Enter<CR>	000 - 000

7.4.2 FINE TUNE SET-POINT COMMAND

The fine tune set-point command allows the user to increment or decrement by one the "on" or "off" parameter of an already existing set-point. The fine tune command is executed by selecting "F6: Fine Tune" from the channel edit menu. When this is done, the fine tune menu is displayed. This menu contains the following commands:

F1: On Setpnt

This is used to select the "on" parameter of the set-point. When selected, the cursor will move to the least significant digit of the "on" parameter.

F2: Off Setpnt

This is used to select the "off" parameter of the set-point. When selected, the cursor will move to the least significant digit of the "off" parameter.

F3: (+) Inc

This command increments the selected parameter (either "on" or "off", whichever was selected with the On Setpnt or Off Setpnt commands) by one.

F4: (-) Dec

This command decrements the selected parameter (either "on" or "off", whichever was selected with the On Setpnt or Off Setpnt commands) by one.

The fine tune command is primarily used in on-line mode to fine tune set-points while the machine is in operation.

7.4.3 PULSE TRAIN COMMAND

The pulse train command is used to program a channel with a fixed “on” and “off” duration through-out the entire channel in one simple command. To program a channel with a pulse train perform the following:

1) Select “F5: Pulse Train” from the channel edit menu.

2) The cursor is located in the “ON:” field of the channel edit menu. Enter the “on” duration in degrees and press Enter<CR>.

3) The cursor is located in the “OFF:” field of the channel edit menu. Enter the “off” duration in degrees and press Enter<CR>.

4) The cursor is now located in the “START:” field. Enter the location that the pulse train will be initially started at in degrees and press Enter<CR>.

5) The entire channel will be programmed with the “on” and “off” duration through-out the entire channel starting at the “start” location. The first 50 set-points (or less) will be displayed in the set-point array.

Note: The set-point cursor will not be displayed since the single set-point and fine tune commands are not valid if the channel is programmed as a pulse train.

SECTION 8
BRAKE WEAR COMPENSATION

The brake wear compensation algorithm can be used in conjunction with presses which incorporate a top dead center (TDC) or back dead center (BDC) stop feature. With these presses, a timing signal is used to declutch the press for TDC or BDC stops. However, as the brake wears, the press will no longer stop at TDC but will instead overshoot. When enabled, the brake wear algorithm of the M4020 will automatically adjust the TDC timing signal such that the press always stops at the desired stopping location regardless of brake wear.

8.1 BRAKE WEAR COMPENSATION PARAMETERS

The brake wear algorithm is implemented in CH00 and can be enabled or disabled in the PLS configuration (see section 7.3.4). When disabled, CH00 functions as a standard timing channel. When enabled, the following parameters, set by the user, are used to implement the algorithm:

Desired Stop Location: This is the location, in degrees, that the press is desired to stop at (i.e. 0 degrees - TDC).

Allowed Stopping Error: This is the plus or minus allowed deviation, in degrees, from the desired stopping location that the press will be allowed to stop before the algorithm adjusts the CH00 timing signal to compensate for brake wear.

CH00 Timing Signal Width: This is the width, in degrees, that the algorithm will program the CH00 timing signal.

Note: The algorithm assumes the leading (off-to-on) transition of the timing signal is used to declutch the press for a TDC stop.

Compensation Enable Window: This is a window, in degrees, in which CH00 is allowed to move in order to compensate for brake wear. This allows the user to place a limit on the amount of compensation the M4020 will perform. If no limit is to be placed on the compensation of CH00, enter “000-000” for the enable window.

In addition to the above parameters, the REMOTE ENABLE input is used to enable the brake wear compensation. The adjustment of CH00 is performed when the press is stopped (RPM = 0) and the REMOTE ENABLE input is “high”. If the REMOTE ENABLE input is “low” when the press is stopped, CH00 is not altered regardless of the stopping location. This allows the algorithm to be by-passed for inch and single modes as well as emergency stops.

SECTION 8
BRAKE WEAR COMPENSATION

8.2 OPERATION OF THE BRAKE WEAR COMPENSATION ALGORITHM

If the brake wear compensation algorithm is to be used, CH00 must be used as the TDC timing signal. CH00 should be wired to an input on the PLC that controls the press, with this input used, in conjunction with other user logic, to declutch the press for TDC stops. The REMOTE ENABLE input on the M4020 must also be controlled such that it is “high” when the press is stopped for TDC stops but is “low” when an emergency (non-TDC) stop is activated or when in any other non-TDC stop clutch mode such as inch or bar is used. It is suggested that a +24VDC sourcing output, from the PLC controlling the press, be wired to the REMOTE ENABLE input, with the appropriate PLC logic controlling this output as described above.

To enable the CH00 brake wear compensation, answer “y” to the “CH00 Brake Wear Compensation enable?” prompt in the PLS configuration (see section 7.3.4). Enter the desired stopping location, allowed stopping error, CH00 timing signal width, and compensation enable window. Initially CH00 must also be programmed with a TDC timing signal. This set-point is used to stop the press after the brake wear compensation is enabled for the first time.

Note: The press may or may not stop at the desired location the first time the brake wear compensation is enabled.

If the press does not stop within the allowed stopping error of the desired stopping location, the algorithm will automatically advance or retard the CH00 timing set-point by the difference between the actual and desired stopping location. At the next TDC stop, the press should than stop at the desired location.

Note: CH00 will only be advanced or retarded within the compensation enable window and no further. If the CH00 set-point is advanced to the leading edge of the enable window, it will not be advanced any further, thus no further compensation for brake wear will occur.

In normal operation, the algorithm verifies that the press always stops within the allowed error of the desired stopping location for all TDC stops (REMOTE ENABLE “high” when press reaches zero speed). As brake wear occurs over time, the press will overshoot the desired stopping location. Once it stops past the allowed error, the algorithm calculates the difference and re-programs the CH00 set-point, advancing it by this difference. The press will than stop at the desired location again. The CH00 set-point is programmed with the width specified in the PLS configuration, however the algorithm assumes the leading (off-to-on) transition is used to actually declutch the press.

8.3 BRAKE WEAR COMPENSATION EXAMPLE

In the this example, the algorithm is implemented on a press which normally requires 180 degrees to stop (1/2 stroke). The parameters are set as follows:

CH00 Brake Wear Compensation Enabled:	yes
Desired Stop Location:	000
Allowed Stopping Error:	+-005
CH00 Timing Signal Width:	040
Compensation Enable Window:	090-270

The following chart shows the automatic reprogramming of the CH00 set-point by the algorithm, as a function of the actual stopping location of the press.

Note: These are not consecutive stops, but instead a sampling of stops over a period of time.

Stop Example	CH00 Set-Point	Actual Stopping Location	New CH00 Set-Point
1	180-220	003	180-220 (no change)
2	180-220	005	180-220 (no change)
3	180-220	007	173-213 (advanced by 7)
4	173-213	359	173-213 (no change)
.	.	.	.
5	173-213	016	157-197 (advanced by 16)
6	157-197	000	157-197 (no change)
.	.	.	.
7	157-197	010	147-187 (advanced by 10)
8	147-187	001	147-187 (no change)
.	.	.	.
9	147-187	054	093-133 (advanced by 54)
10	093-133	003	093-133 (no change)
.	.	.	.
11	093-133	011	090-130 (fully advanced)
12	090-130	008	090-130 (fully advanced)
.	.	.	.
13	090-130	032	090-130 (fully advanced)
14	090-130	035	090-130 (fully advanced)
.	.	.	.
15	090-130	311	139-179 (retarded by 49)
16	139-179	352	147-187 (retarded by 8)
17	147-187	358	147-187 (no change)

In all cases above, normal TDC stops were performed with the REMOTE ENABLE input “high”.

Note: In the above chart, when the stopping location was within the allowed error (+-5 degrees or less) no change was made in the set-point. Also, once the set-point was set to 90-130 that no further compensation occurred, even though the press still stopped outside the allowed error. This is because the compensation window was set at 90-270 degrees. The algorithm only programs the CH00 set-point within the compensation window. If no limit on the amount of compensation is desired, simply set the compensation window to 000-000. This allows CH00 to be reprogrammed anywhere.

SECTION 9
RPM/POSITION OUTPUT CONNECTOR

In addition to the four digit RPM/POSITION display built into the M4020, an RPM/POSITION output connector is provided to drive a remote RPM/POSITION display or for interfacing to a PLC, allowing the PLC to read the RPM or POSITION.

9.1 RPM/POSITION OUTPUT CONNECTOR DESCRIPTION OF OPERATION

The connector contains eight outputs and two inputs. The eight outputs are 24VDC sinking (true low) outputs which multiplex the data for the four digits using four outputs for the digit data and four outputs for the individual digit strobes. Figure 9.1 is a timing diagram of a typical remote display update. For each digit (1's, 10's, 100's and 1000's), the digit data is placed on the digit outputs (1's, 2's, 4's, and 8's) and held for a data "set-up" time. The respective strobe output (STRB DGT0, 1, 2, 3) is then activated "low" for a data "strobe" time. The strobe is then released and the digit data is held for a data "hold" time.

Note: The digit data is always valid when the digit strobe is "low", this is when the digit data lines should be sampled and read-in. Each digit of the remote display is updated, starting with the 1's digit, until all digits are updated, at which time the entire process is then started over. The remote display is thus updated automatically and continuously this way.

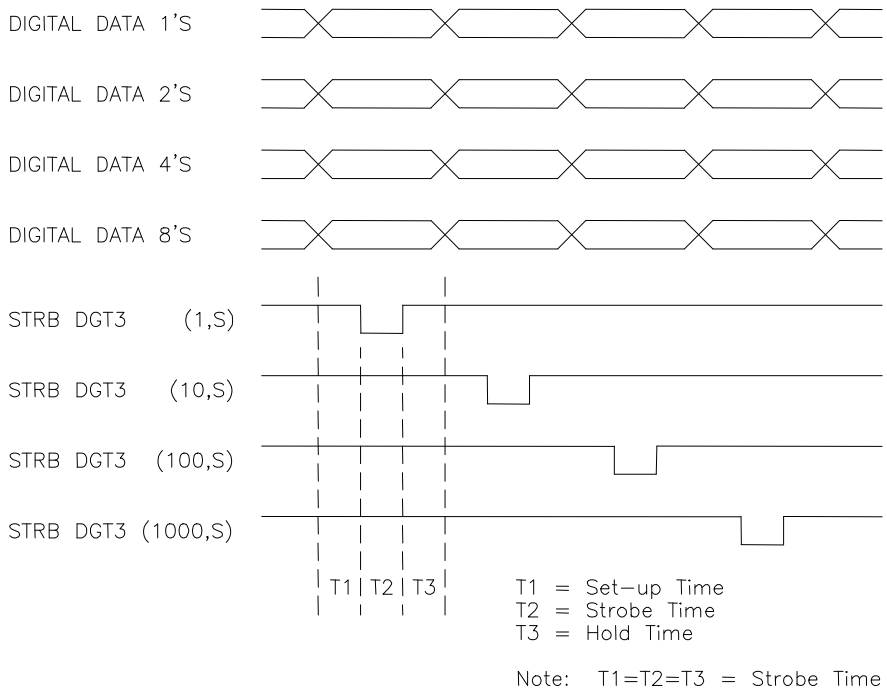


Figure 9.1 – Typical Remote Display Update

SECTION 9
RPM/POSITION OUTPUT CONNECTOR

The two inputs on the RPM/POSITION output connector are 12-30VDC sourcing (true high) inputs labeled REMOTE ENABLE and RPM SELECT. If REMOTE ENABLE is “low”, RPM or POSITION is selected locally via the RPM/POSITION toggle switch on the M4020. If REMOTE ENABLE is “high”, POSITION is displayed when RPM SELECT is a “low” and RPM is displayed when RPM SELECT is a “high”. When REMOTE ENABLE is “high”, the strobe time of the digit strobe outputs can be user programmed from 2.5msec to 640msec in 2.5msec increments. This allows the user to match the output strobe time to the device the remote output connector is interfaced to. If REMOTE ENABLE is “low” (no remote display connected), the digit strobe times are in microseconds and would be ineffective in driving a remote device. Thus if a remote device is connected to the M4020, REMOTE ENABLE should be “high”. The strobe time is set in the PLS configuration (see section 7.3.3).

Note: When REMOTE ENABLE is “high”, the RPM/POSITION toggle switch on the M4020 has no effect, either on the remote display or built-in display. Also, the data placed on the digit data lines is the binary equivalent of the digit number as shown in the following table:

Digit Number	Binary Equivalent 8's	4's	2's	1's
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
blank	1	1	1	1

When interfacing with a PLC, the "blank" digit (digit data equal to all "1"s) should be converted to zero by the PLC logic.

9.2 REMOTE RPM/POSITION DISPLAY INTERFACE

Figure 9.2 shows a Cincinnati Electro Systems 4161-4-24 BCD display wired to the M4020 RPM/POSITION connector. This would be a typical remote display application.

Note: The REMOTE ENABLE is tied "high" and that a switch, located at the remote display, is wired to the RPM SELECT input. This switch allows the remote selection of either RPM or POSITION to be displayed. In this example, the strobe time could be set as low as 2.5msec (or the required strobe time of the display).

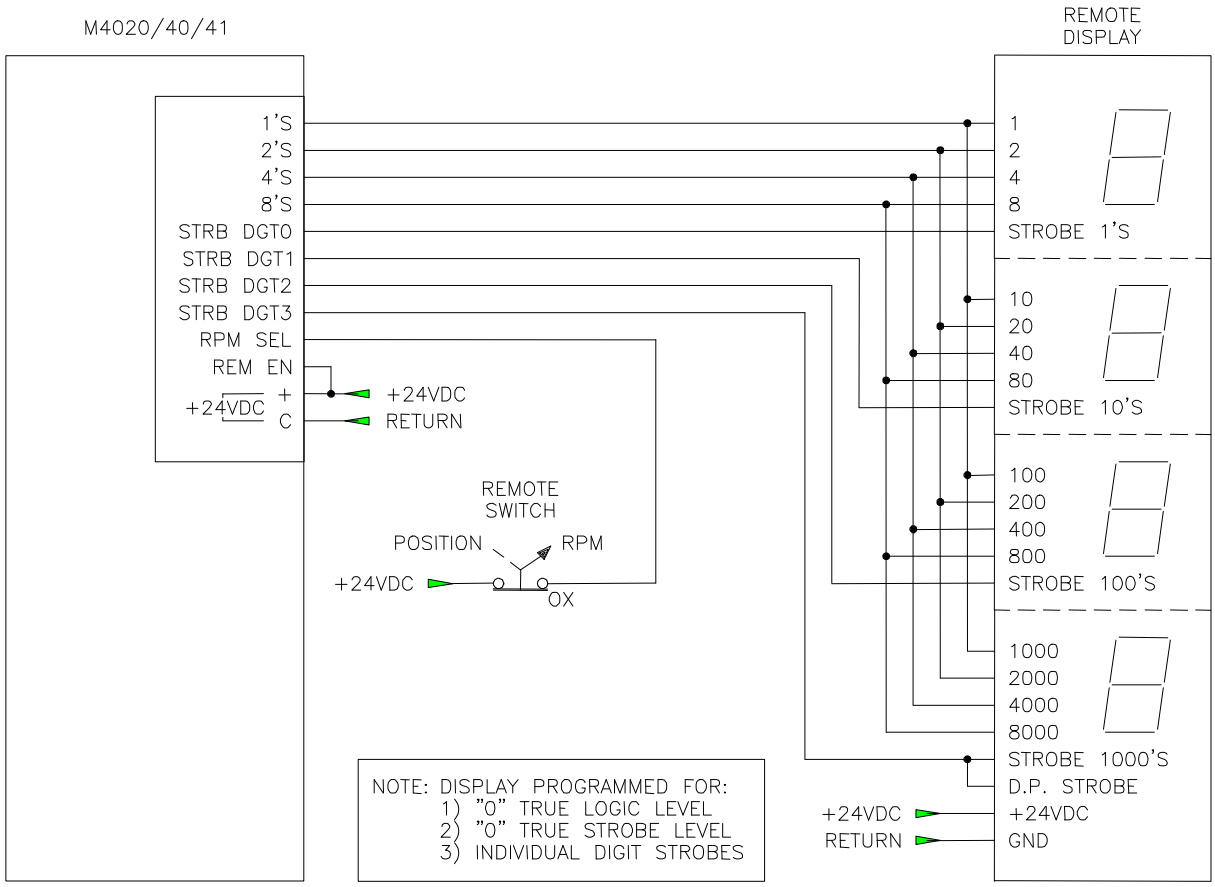


Figure 9.2 – RPM/Position Output Connector Wired to Remote Display

SECTION 9
RPM/POSITION OUTPUT CONNECTOR

9.3 INTERFACING THE RPM/POSITION OUTPUT TO A PLC

Figure 9.3 shows the M4020 RPM/POSITION output interfaced to a typical PLC (Programmable Logic Controller). This is an example of how a PLC can read the RPM or POSITION from the M4020. The digit data and strobe outputs of the M4020 are wired to PLC inputs while the REMOTE ENABLE input is tied “high” and RPM SELECT input is wired from a PLC output. This allows the PLC to read either the RPM or the POSITION. In this case the digit strobe time would have to be set to a time greater than or equal to the following:

strobe time >= PLC input filter delay time (max) + (worst case PLC scan time X 2)

Setting the strobe time to the above guarantees that each digit will be read by the PLC. The PLC logic would than clock in the data for each digit when the corresponding digit strobe is “low”. Once the 1000’s digit strobe occurs, the equivalent RPM or POSITION (whichever is selected by the PLC) can be determined from the individual digits strobed in.

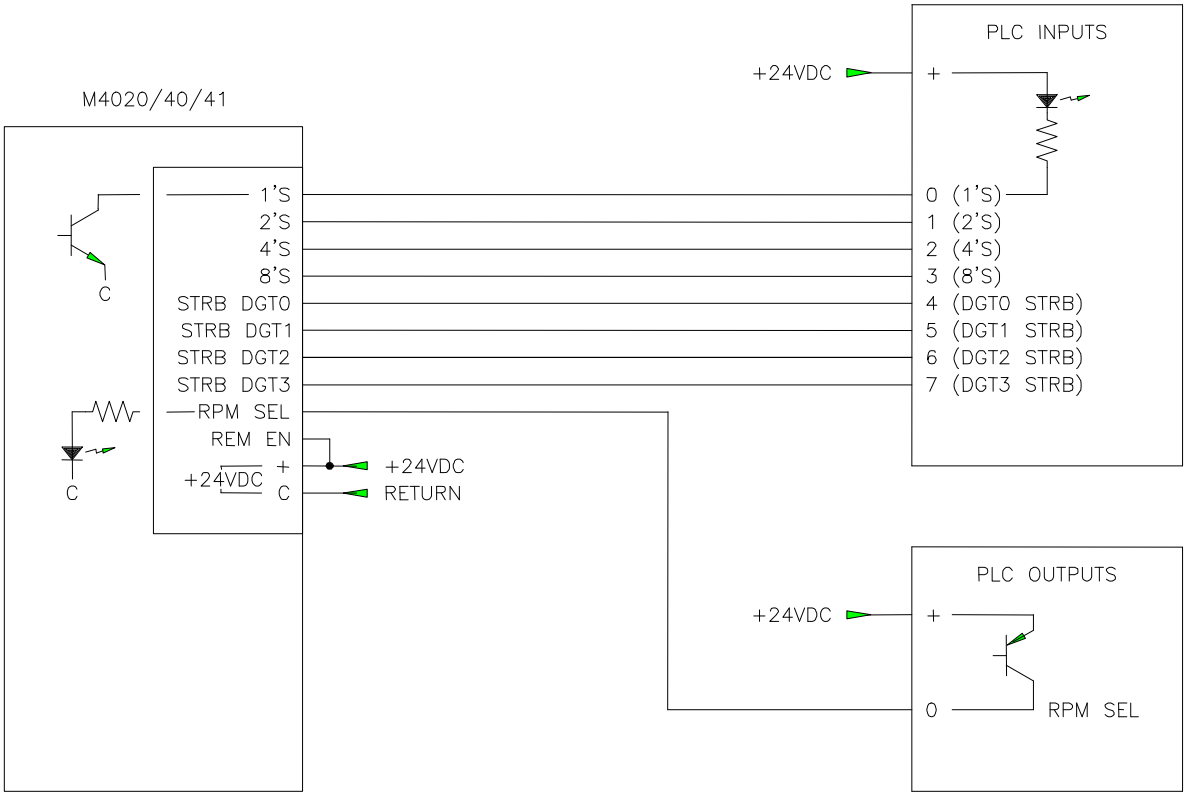


Figure 9.3 – RPM/Position Output Connector Wired to PLC

SECTION 10
FAULT DETECTION

The M4020 contains comprehensive fault detection routines which verify the proper operation of the module at all times. If the module detects a fault condition, the “FLT” LED on the front of the module is illuminated and the fault routine is executed. The sources of these faults range from a hardware failure of the module to an error in the user’s program (infinite loop, etc.).

10.1 FAULT ROUTINE EXECUTION

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

1) User program execution is suspended.
2) If possible, all outputs in the system are disabled
3) ”FLT” LED on the front of the module is illuminated.
4) ”RUN” LED is extinguished.
5) Fault interlock is opened
6) Fault code representing the detected fault is saved in internal memory of the module for viewing with SYSdev.

The first step in correcting a fault condition (FLT LED “on”) in the M4020, is viewing the fault code saved inside the module with SYSdev.

10.2 VIEWING FAULT CODES WITH SYSDEV

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

1) Connect IBM PC “COM1” port to M4020 “PROG” port using the appropriate cable (see appendix B).
2) Initiate SYSdev from the DOS prompt and select the user program currently loaded in the module.
3) From the main menu, select “Target Board Interface”.
4) From the Target Board Interface menu, select “Target Board Fault Codes/Status”.

SECTION 9
RPM/POSITION OUTPUT CONNECTOR

The SYSdev fault display reads the fault codes from the module 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:

Curr Flt: This is the M4020 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 M4020 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 module 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 M4020 module.

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.

10.3 FAULT CODES

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

Code	Description
00H	No internal fault has occurred
40H	Watchdog timer 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

10.3.1 WATCHDOG TIMER TIMEOUT (40H)

The watchdog timeout fault occurs when the main program scan time exceeds 100 milliseconds. The cause of this fault ranges from an error in the user program(unintentional loop entered in the user program, unintentional indirect access to program memory) to a hardware failure of the M4020 module.

Trouble-shooting:

1) Check the program for any unintentional 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 jump.

2) Check for any loop instructions that may take longer than 100 milliseconds to execute (a large number of iterations through the loop).

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

4) If the problem persists, try another M4020 module to verify if a hardware problem exists.

10.3.2 IBM PC TO M4000 COMMUNICATIONS FAILURE (42H)

If an attempt to read the fault codes from the M4020 module results in an error code of “42H: Cannot communicate with target board”, the PC cannot communicate with the module. This is not an internal M4020 fault but instead a fault detected by SYSdev. The cause of this fault ranges from catastrophic failure of the module to a misconnection of the PC to the module.

Trouble-shooting:

1) Verify the “PWR” LED on the module is on. If not, verify that +24VDC power is applied to the module.

2) Verify that the RS-232 cable is connected to “COM1” on the PC and “PROG” port on the module.

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

4) If the above verifies, replace the M4020 module and try again. If the problem still persists, verify the “COM1” port for proper operation (see manual from PC manufacture).

10.3.3 INVALID PROGRAM FAULTS (5CH and 5DH)

The “Program invalid” (5CH) fault occurs when the module does not contain a valid user program. This typically occurs when a new module 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 M4020 module is interrupted while program download is in progress.

Trouble-shooting:

1) Dump the user program to the M4020 module. These faults will clear once the module is loaded with a valid user program.

2) If re-loading the module with the user program does not clear the fault, replace the M4020 module and try again.

10.3.4 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 5.2.5 for details on sfunc09.

10.3.5 INTERNAL M4020 FAULTS (43H,44H,59H-5BH)

The remainder of the fault codes detected by the M4020 module represent an internal failure of the module. These can range from the RAM battery low to invalid interrupt requests.

Trouble-shooting:

1) Perform the hardware confidence test on the M4020 module. It may be desirable to remove the suspect module from the system and to install another module to get the application being controlled back up and running. See section 11 for details on the test.

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

10.4 SERIAL NETWORK COMMUNICATION ERRORS

Unlike the system faults, the serial network communication errors do not cause the M4020 module to 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 10.2 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.

10.4.1 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
09H	Receive alignment error (bad frame)
0AH	Receive CRC error
0BH	Unknown (undefined) error
0CH	Transmit no acknowledge
0DH	Transmit underrun error
0EH	Transmit collision detected
0FH	Address error (outside data memory)
10H	Unexpected slave responding

10.4.2 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 12.7.2.

3) If the problem persists, replace the slave M4000 module where the problem is occurring. Next replace the M4000 master module.

10.4.3 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 slave M4000 module with which the error occurred. If the problem persists, replace the M4000 module at the master node.

10.4.4 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.

10.5 PLS SECTION FAULT CODES

The PLS section detects three fault states. These are: invalid scale factor (code 07), invalid offset (code 08), and timing channel output short circuit (code 09). When the M4020 detects any one of these faults, the respective fault code is displayed on the RPM/POSITION display of the module in the form “88XX” where XX is the fault code. The individual faults are discussed below:

10.5.1 8807: INVALID SCALE FACTOR

This fault is displayed when the scale factor is greater than 512 or less than 2. To reset the fault, download the user channel program through PLSdev. This downloads the configuration file which contains the scale factor as specified by the user.

10.5.2 8808: INVALID OFFSET

This fault is displayed any time the offset is greater than or equal to the scale factor. To reset the fault, set the offset to less than the scale factor by downloading the user channel program or by setting the offset in the on-line mode of PLSdev.

10.5.3 8809: TIMING CHANNEL OUTPUT SHORT CIRCUIT

This fault is displayed anytime any one of the timing channel outputs detects a short circuit (current greater than 500mAMP) even if only momentarily. Once the short circuit condition is removed, the fault can be reset by toggling the RPM/POSITION toggle switch on the front of the M4020.

SECTION 11
HARDWARE CONFIDENCE TEST

The hardware confidence test allows the entire M4000 module hardware to be verified for proper operation. The test is resident in all modules and is initiated through SYSdev and PLSdev. The hardware confidence test is the same test used at the factory to initially test the production M4020 modules, and therefore provides the same 100% hardware test as provided at the factory.

The test is provided to the user to verify whether the module hardware is functional or not. Not as a tool to repair the modules. If a fault is detected, the module should be returned to the factory for repair. Any attempt to repair an M4020 module will void the warranty.

11.1 TESTS PERFORMED - PLC SECTION

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

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

Tests 1, 3, and 4 are not optional and are always performed. Test 2 is normally disabled but can be enabled if desired.

Note: If test 2 is to be performed, the FLT interlock output must be wired to the '-' terminal of both the interrupt input0 and input1 inputs. The '+' terminals of both interrupt input0 and input1 must be wired to the '+' terminal of the power input (+24VDC). Test 2 uses these two inputs to verify the FLT interlock output. Failure to connect these inputs as described will result in a fault detected when test 2 is performed. Test 5 is optional and may be disabled if desired. All tests are automatic and require no interaction once the test is initiated.

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

Note: The actual input and output points hardware is not checked with these tests. This can be done using the on-line monitoring mode of SYSdev to view the states of the inputs and set the states of the outputs.

SECTION 11
HARDWARE CONFIDENCE TEST

11.2 TESTS PERFORMED - PLS SECTION

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

1) Micro controller RAM test
2) Channel 00-07 PLS memory test
3) Channel 10-17 PLS memory test
4) Channel 20-27 PLS memory test
5) Channel 30-37 PLS memory test
6) PLS Memory address access test
7) RS-232 Chan port test

Tests 1 through 6 are not optional and are always performed. Test 7 is optional and may be disabled if desired. All tests are automatic and require no interaction once the test is initiated.

Each test performs a complete check of the respective hardware area of the M4020. If a fault is detected, the test is stopped and test fault code is displayed to indicate the nature of the hardware failure.

11.3 PERFORMING THE PLC HARDWARE CONFIDENCE TEST

WARNING: The hardware confidence test should not be performed in an M4020 module installed in a user's control system. Unpredictable output states may result while the test is being performed.

11.3.1 EQUIPMENT REQUIRED

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

1) IBM PC or compatible with SYSdev installed.

2) RS-232 interface cable to connect "COM1" on the PC to "PROG" port on the M4000 module.

3) +24VDC power supply to power module.

4) M4020 module to be tested.

11.3.2 EXECUTING THE PLC TEST

To execute the test, perform the following steps:

1) Power up the M4020 module to be tested.

2) Power up PC and enter SYSdev. (If using the SYSdev shell, select "M4020 (PLC)", to invoke SYSdev.) Enter any user program name to proceed to the SYSdev Main Development Menu.

3) Connect Interface cable to “COM1” on PC and “PROG” port on module.

4) Select “Target Board Interface” from the Main Development Menu then select “Target Board Hardware Confidence Test” from the Target Board Interface menu.

5) Select “M4000 Confidence test” from the confidence test menu. A prompt will be displayed verifying to proceed with the test.

   Note: Proceeding with the test will clear the program and data memory in the module. The user application program will have to be re-downloaded to the module once the test is complete. Press “ESC” to abort the test, any other key to proceed.

6) Select “Perform Test” from the Test Functions Menu 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 Development Menu. The user application program will now have to be re-downloaded to the M4020 module.

SECTION 11
HARDWARE CONFIDENCE TEST

11.4 PERFORMING THE PLS HARDWARE CONFIDENCE TEST

WARNING: The hardware confidence test should not be performed in an M4020 installed in a users control system. Unpredictable output states may result while the test is being performed.

11.4.1 EQUIPMENT REQUIRED

In order to perform the M4040/41 hardware confidence test, the following is required:

1) IBM PC or compatible with PLSdev installed.
2) RS-232 interface cable to connect “COM1” on the PC to “CHAN” port on the M4020.
3) +24VDC power supply to power the M4020.
4) M4020 to be tested.

11.4.2 EXECUTING THE PLS TEST

To execute the test, perform the following steps:

1) Power up the M4020 to be tested.
2) Power up PC and enter PLSdev (if using the SYSdev shell, select “M4020 (PLS)”, to invoke PLSdev). Enter any user program name to proceed to the PLSdev Main Development Menu.
3) Connect Interface cable to “COM1” on PC and “CHAN” port on M4020.
4) Select “PLS Hardware Confidence Test”. A prompt will be displayed verifying to proceed with the test.

   Note: Proceeding with the test will clear all set-points and configuration data from the M4020. The user channel program will have to be re-downloaded to the M4020 once the test is complete.

   Press “ESC” to abort the test, any other key to proceed.
5) Select “Perform Test” from the Test Functions Menu to start the test. Once the test is initiated, all tests enabled will be executed repeatedly, starting with test 1 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.

11.5 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 RS-232 ports, to any address in the module. In general, all these selections are for factory use and are of little significance to the user.

SECTION 11
HARDWARE CONFIDENCE TEST

(This Page Intentionally Left Blank)

SECTION 12
INSTALLATION

The following sections provide information on mounting and wiring the M4020 module as well as a description of the power-up sequence.

Note: All wiring is implemented with removable field wiring connectors. The connectors are removed by gently pulling the connectors from the socket. Install the connectors by firmly seating the connector to the socket, observing the proper polarity of the connector. Refer to appendix C for the pin-outs of the various connectors on the M4020 module.

12.1 MOUNTING THE M4020

The M4020 module was designed for back panel mounting. The module should be mounted using 2ea. 8-32 screws and lock washers. A lugged earth ground wire should be installed on one of the mounting screws to insure that the module is grounded.

12.2 WIRING INPUT POWER

The M4020 module is powered with +24VDC, +-10% power. This power is wired to the '+' (power) and 'C' (common or return) terminals of the input interrupt connector and to the '+' (power) and 'C' (common or return) terminals of the RPM/POSITION output connector.

Note: Both sets of these power input terminals must be wired to provide power to both the PLC and PLS sections of the M4020. This provides power to the internal circuitry of the module. The current required is less than 0.8 AMPs. Be sure to observe the proper polarity of the input power, otherwise damage to the module may occur. Input fusing of 2AMP or so should be provided for the input power.

12.3 WIRING 10-30VDC DIGITAL INPUTS

The digital inputs are 10-30VDC sourcing (true high) inputs which are used to interface to sourcing application inputs such as proximity sensors, push-buttons, etc. The inputs are internally mapped to terminals on the input connector numbered with the corresponding input number. All inputs are commoned to the 'C' (common or return) terminal of the connector.

Note: The inputs are optically isolated, thus the common of the input voltage does not have to be commoned with the +24VDC power used to power the module. Figure 12.1 shows typical input wiring.

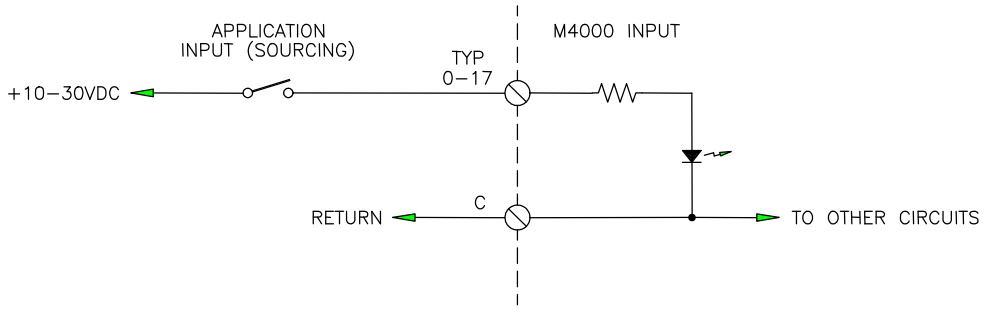


Figure 12.1 – Typical M4000 Input Wiring

12.4 WIRING INTERRUPT INPUTS

Interrupt input0 and input1 are 12-30VDC differential inputs which can be wired as sourcing (true high), sinking (true low), or as true differential inputs (driven by a differential output). Each input is provided with a '+' and '-' terminal. Figure 12.2 shows wiring examples of all three types of input configurations.

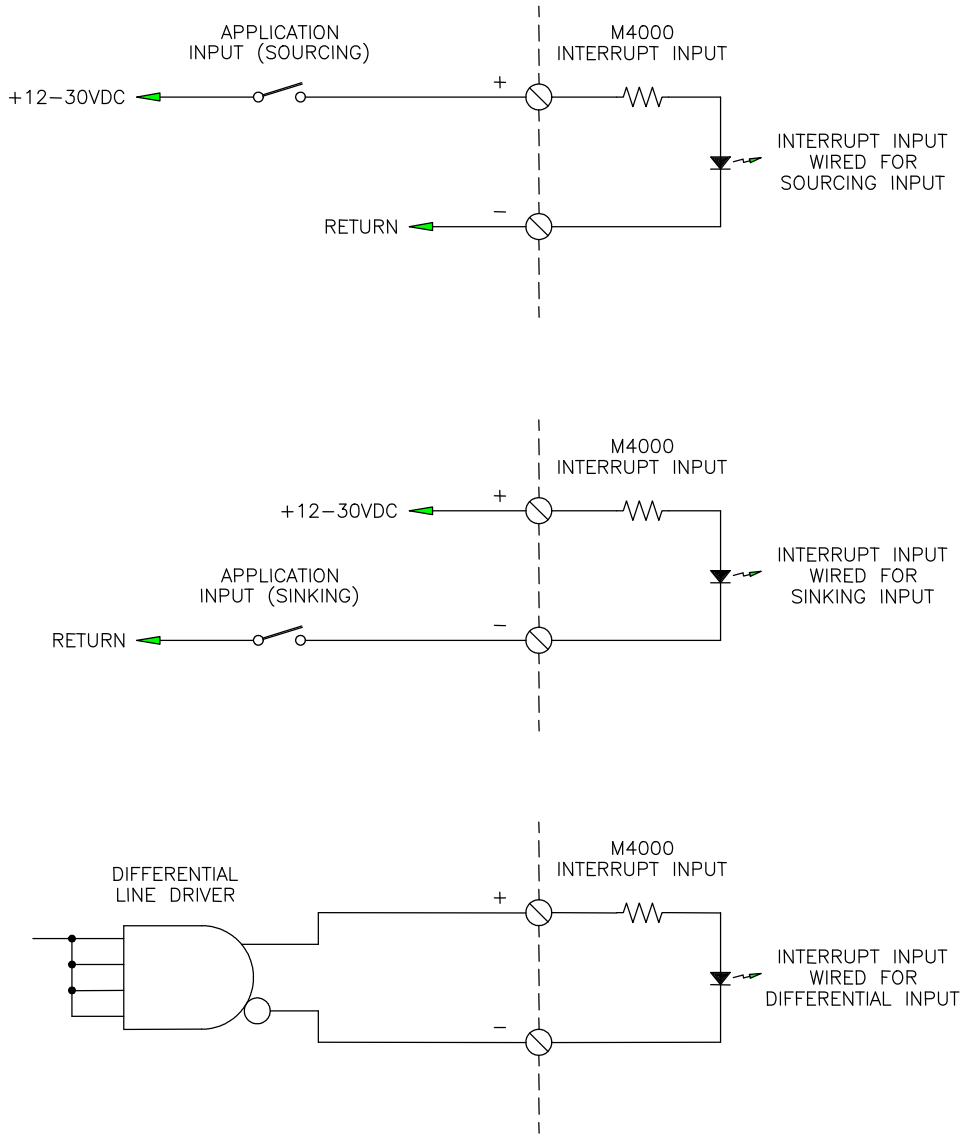


Figure 12.2 – Typical Interrupt Input Wiring

12.5 WIRING 10-30VDC DIGITAL OUTPUTS

The digital outputs are 10-30VDC sourcing (true high) which are used to interface to the application outputs such as solenoids, lamps, PLC inputs, etc. Each output is rated at 1 amp DC (continuous) with an inrush (pulsed) current drive capability of 5 amps for 100msec. The outputs do not contain output fusing or short circuit protection, therefore external fusing should be provided. Power for the digital outputs is wired to the '+' (power) and 'C' (common or return) terminals on the output connector. Be sure to observe the proper polarity of the '+' power and 'C' wiring, otherwise damage to the module may occur.

Note: The outputs are optically isolated, thus the 10-30VDC power applied to the output connector does not have to be commoned with the +24VDC power used to power the module. Figure 12.3 shows an example of the typical output wiring.

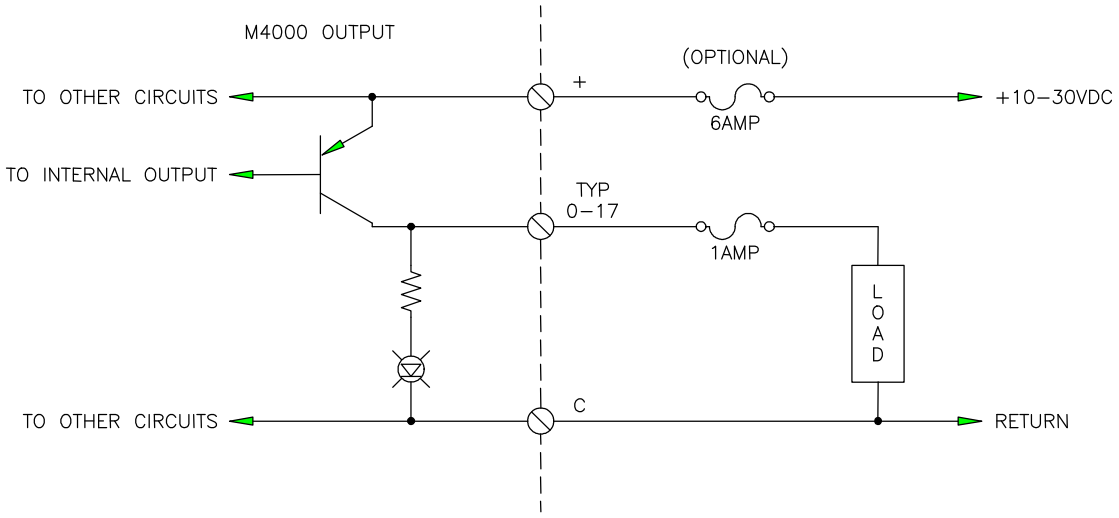


Figure 12.3 – Typical Output Wiring

12.6 WIRING THE FAULT INTERLOCK

The fault interlock is a +24VDC sinking (true low) output which can be interfaced to an external relay or PLC input to indicate a fault condition with the M4020 module. The output is capable of sinking 100 milliamps. The fault output is “on” (true low - sinking current) when the module is executing the user program properly. If a fault condition is detected, the fault output is turned “off” (high). Figure 12.4 shows the fault output wired to a +24VDC relay. This relay could be interlocked with the digital outputs power to remove power from the outputs if the module was to fault out.

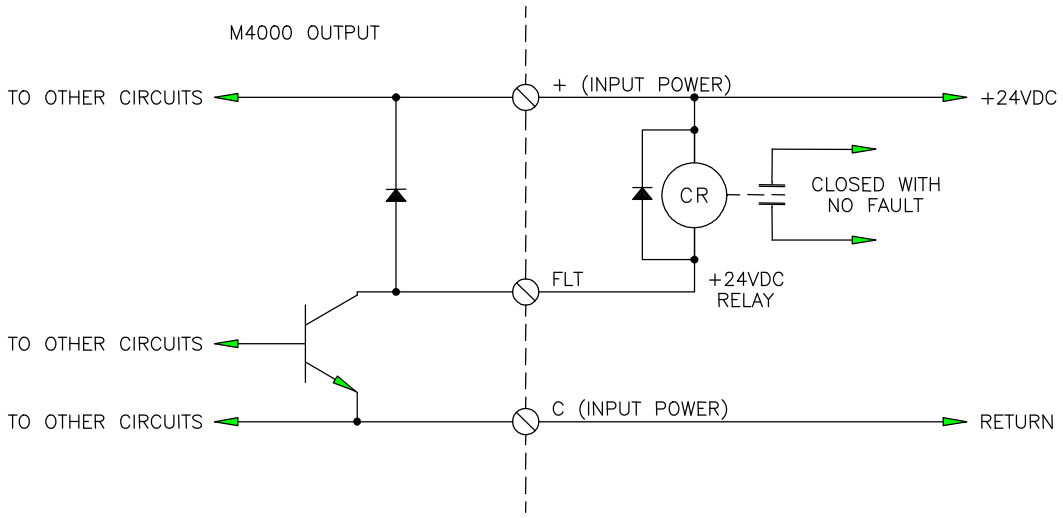


Figure 12.4 – Typical Fault Interlock Wiring

12.7 SERIAL NETWORK INSTALLATION

The serial network installation consists of wiring the network and setting each M4000 module on the network with a unique network address. Up to 32 M4000 modules can be installed on one network.

12.7.1 WIRING THE SERIAL NETWORK

Refer to figure 12.5 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:	0.004 db/ft.
Twisted pair, single-shielded	

2) The total wire length of the network cannot exceed 1000 ft. if the network baud rate is set to 344KBPS, 2000 ft. for 229KBPS, and 4000 ft. for 106KBPS. See section 3.2 for details on setting the network baud rate.

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 12.5.

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 module (+ 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 module. However, if a connector is pulled from it's module, communications to the modules 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 12.6.

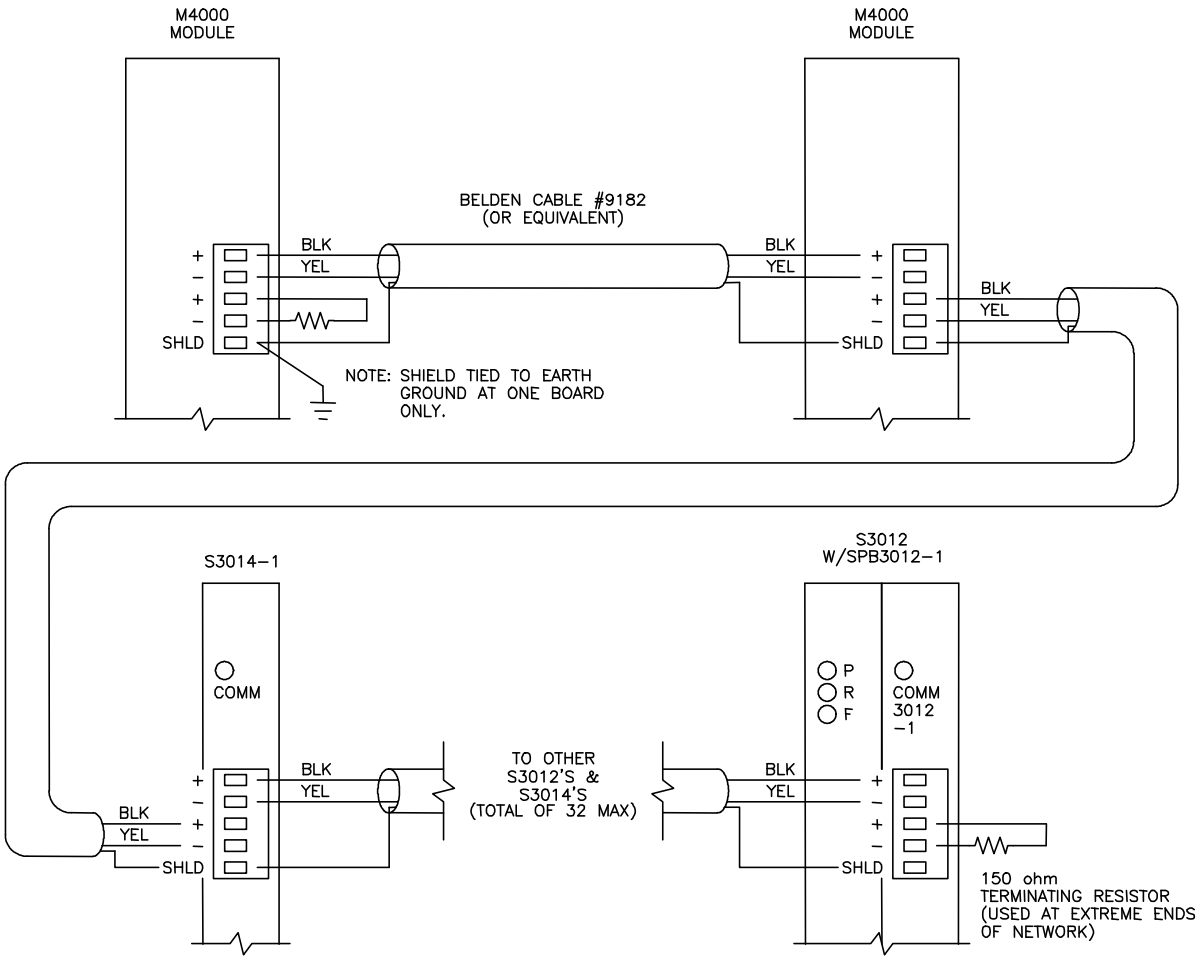


Figure 12.5 – Typical Network Wiring

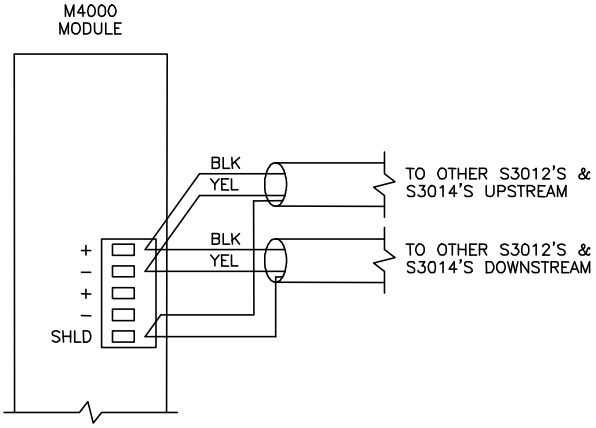


Figure 12.6 – Alternative Serial Connector Wiring

12.7.2 SETTING THE NETWORK ADDRESSES

Each M4000 module on the network must be set with a unique network address between 1 and 32. This is how the modules can distinguish one node from another. To set the network address for a particular module, perform the following:

1) Connect an IBM PC or compatible running SYSdev from “COM1” on the PC to “PROG” port on the module 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 M4000 module and display it in the network display. If the network address is to be changed, follow the directions displayed and enter the new address.

The above steps must be done for all M4000 modules on the network. This is true when the network is first installed, and when a new module is added or replaced (that module must have the network address set it in).

12.8 RESOLVER INTERFACE

The M4020 module can be used with virtually any type of resolver which incorporates a rotor reference signal and two stator feedback signals. These include resolvers manufactured by C&A, Autotech, Gemco, etc. A dip switch accessible through an access hole on the left side of the module selects the desired resolver reference voltage, either 1.45VRMS or 3.70VRMS. Separate resolver reference input and output connections are provided on the resolver interface connector. This allows the M4020 module to be slaved to other M4000 PLS modules or to other manufactures programmable limit switches off one resolver.

12.8.1 RESOLVER WIRING

Figure 12.7 shows the typical wiring to a generic resolver for stand alone operation (M4000 PLS provides reference voltage for resolver).

Note: The reference output (RO) is tied to the reference input (RI). The reference output excites the resolver while the reference input and two stator inputs are used to determine the absolute position of the resolver. The direction of rotation is determined by the connection S1 and S3. Swapping S1 and S3 will reverse the direction of rotation for the M4000 module.

Figure 12.8 shows two M4000 modules wired to one resolver with module #2 slaved to module #1 (reference for resolver supplied by module #1). In this case the reference output (RO) of module #1 is connected to the resolver and tied to the reference inputs (RI) of both modules while the reference output of module #2 is not used. Up to 8 modules can be slaved off one resolver this way.

In all cases, a shielded twisted pair cable should be used to wire the resolver to the M4020, making sure the cable is routed free of other high voltage wiring. The shield connection (SH) should be tied to either the resolver reference ground (R2) or to earth ground.

12.8.2 RESOLVER REFERENCE VOLTAGE SELECTION

The resolver reference voltage of the M4020 module can be selected for either 1.45VRMS or 3.70VRMS depending on the requirements of the resolver. For Autotech and Gemco resolvers, the reference should be set for 1.45VRMS. For C&A resolvers, the reference should be set for 3.70VRMS. For other resolvers, refer to the manufactures specifications to determine which reference should be selected.

The reference is selected via dip switches accessible through an access hole on the lower left side of the module. To select the 1.45VRMS reference, set both switches down on the side that reads “1.45VRMS”. To select the 3.70VRMS reference, set both switches up on the side that reads “3.70VRMS”.

Note: The proper reference voltage level must be selected for the specific resolver used, otherwise erroneous operation may result.

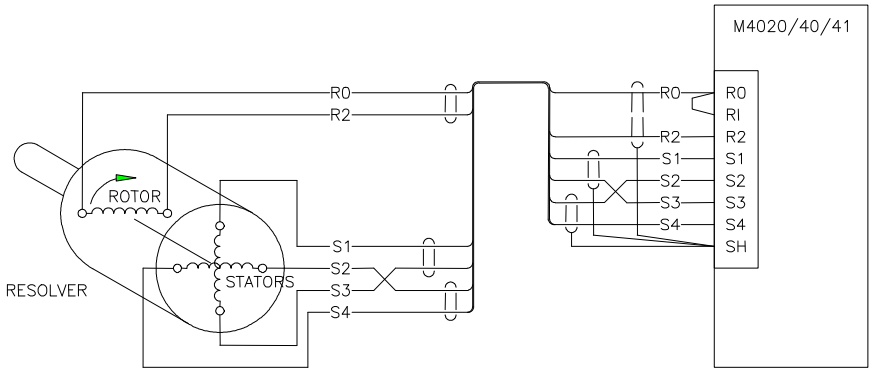


Figure 12.7 – M020/40/41 Resolver Interface (Stand Alone)

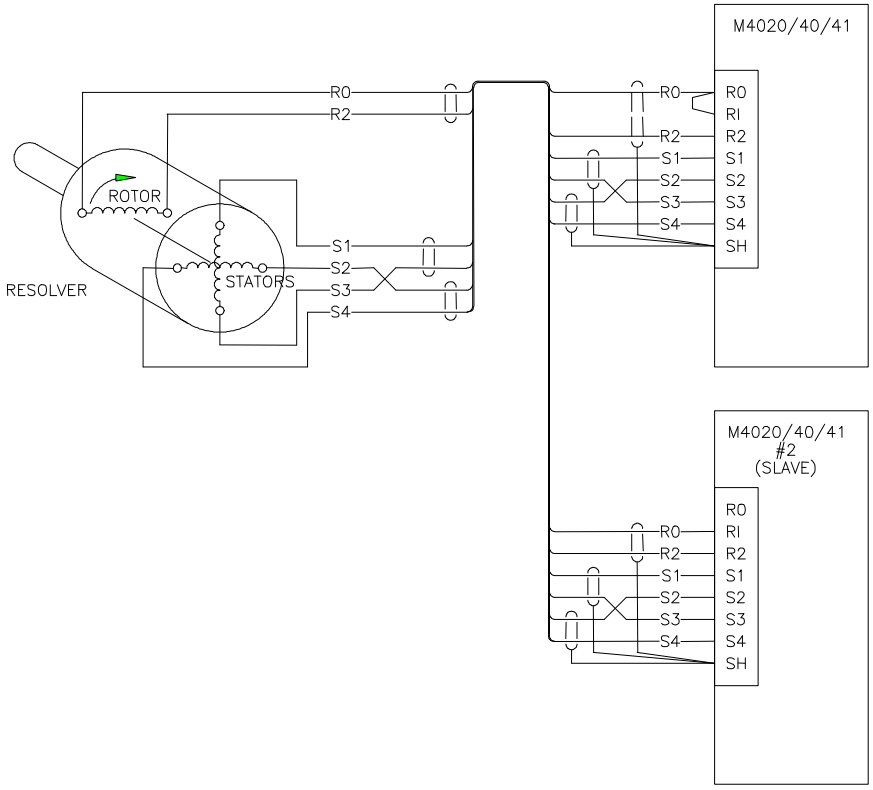


Figure 12.8 – M4020/40/41's Slaved to One Resolver

12.9 WIRING THE RPM/POSITION OUTPUT CONNECTOR

The RPM/POSITION output connector is provided to drive remote RPM/POSITION displays or to interface with a PLC. The connector consists of eight 10-30VDC true low outputs capable of sinking 100milliamps and two 12-30VDC sourcing inputs. See section 9 for examples of wiring this connector.

12.10 WIRING TIMING CHANNEL OUTPUTS

The timing channel outputs are 10-30VDC true high outputs capable of sourcing 100milliamps of current. The outputs were designed to drive PLC inputs or other low current devices and are not capable of driving solenoids. The timing channel outputs contain short circuit protection such that if the output current exceeds 500milliamps, all outputs within the eight channel group are disabled, protecting the output. Power for the timing channel outputs is wired to the '+' (power) and 'C' (common or return) terminals on the Timing channel output connector. Be sure to observe the proper polarity of the '+' power and 'C' wiring, otherwise damage to the module may occur.

Note: The timing channels are optically isolated, thus the 10-30VDC power applied to the timing channel outputs does not have to be commoned with the +24VDC input power described above. Figure 12.9 shows the typical timing channel output wiring.

12.11 WIRING THE 128PPR AND 1KPPR OUTPUTS

The 128 and 1K pulse per revolution outputs are implemented with an MM88C30 differential line driver. These outputs provide fixed pulse train signals that can be used to drive digital tachometers, etc. The devices these outputs drive must incorporate differential inputs (do not ground either output). See figure 12.10 for a typical example of the 128PPR and 1KPPR output wiring.

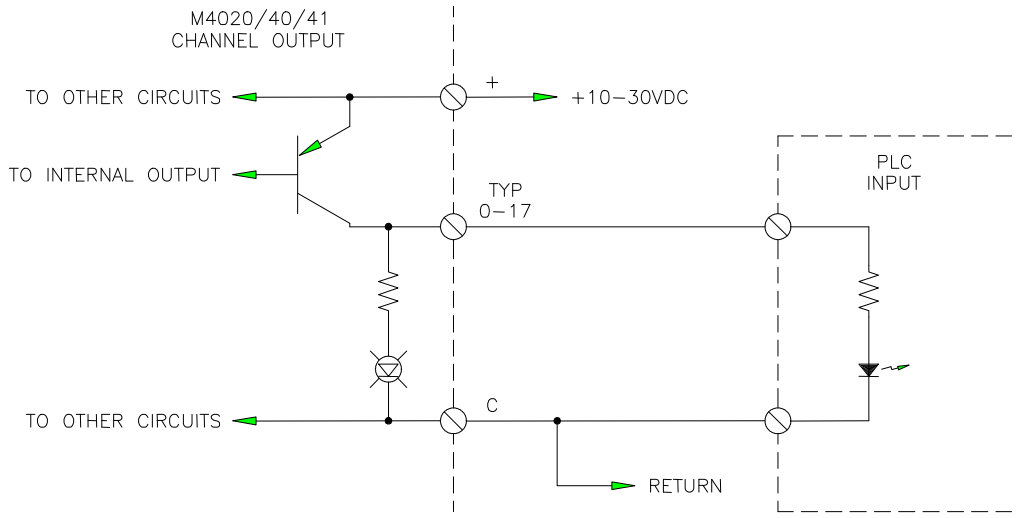


Figure 12.9 – Typical Channel Output Wiring

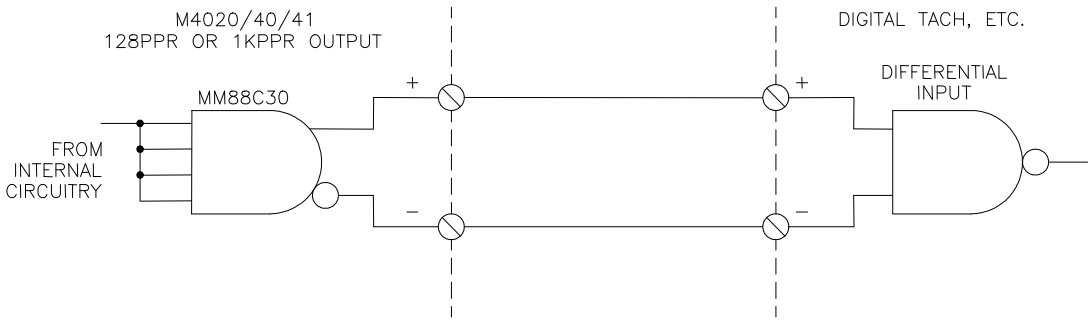


Figure 12.10 – Typical 128ppr/1Kppr Output Wiring

12.12 POWER-UP SEQUENCE OF M4020 MODULE

Once all connectors are wired and re-installed in their respective sockets, apply +24VDC power to the module. The power-sequence occurs as follows:

1) At initial power-up, the module is reset for approximately half a second. During this time, the fault interlock will be “off”, and the “FLT”, “RUN” and “PWR” LEDs will all be “on”.

   Note: During this time, the outputs of the module may also be “on” or “off” randomly. This is because the processor of the module has not started execution of the program yet and thus has no control of the outputs. If this is a problem, power for the outputs can either be supplied to the outputs from a separate source which is not activated until after the module is powered up or by using a time delay relay which supplies power to the outputs a time delay (one second or so) after power is applied to the module. During the reset cycle, the RPM/POSITION display will show arbitrary data (decimal points, arbitrary digits, etc.). The timing channel outputs may also be randomly “on” or “off”. This is normal.

2) Once the reset cycle is complete, the module will begin to execute the program previously loaded. The fault interlock will turn “on” (sink true low) and the “FLT” LED will extinguish. The outputs will then be activated in the states as controlled by the user program. If the timing channels have previously been programmed, the “offset” value previously set will be displayed for approximately 2 to 3 seconds. The module will then proceed to normal operation with the RPM/POSITION display showing either the RPM or position, whichever is selected.

3) If a user program has not been loaded (new module or module which the confidence test has just been executed), the “FLT” LED will stay “on” and the “RUN” LED will extinguish. Download the user program and data files to the module. The “RUN” LED will flash while the user program is downloading. When the download is complete, the “FLT” LED will extinguish and the “RUN” LED will turn “on”. If the “FLT” LED turns “on” after the download is complete, read the fault code in the M4000 module (see section 10). See the SYSdev program manual for details on downloading programs to the M4000 modules. If the timing channels have not been previously programmed (new module), it may show either the invalid scale factor fault code (8807) or invalid offset fault code (8808) on the RPM/POSITION display, indicating that the scale factor and/or offset has not been set yet. Download the configuration and channel data to the M4020 using the “4: Download Channels to PLS” selection of PLSdev. See section 7 for details on setting the configuration, programming the channels, and downloading the channels to the M4020.

SECTION 12
INSTALLATION

(This Page Intentionally Left Blank)

APPENDIX A
PROGRAMMING EXAMPLE

The following is an M4020 program example. The program contains various examples of ladder and high-level blocks. The name of the program is “M4020E1” and it was copied into a directory named “EXAMPLES” (which is a sub-directory of SYS51) by the installation program when SYSdev was installed on your hard drive. To view the program in SYSdev, perform the following:

1) From the root directory of the drive you installed SYSdev on, type SYSDEV and press ENTER. SYSdev will be invoked, displaying the directories and programs in the root directory of the current drive.

2) Select the “SYS51” directory using the “F3: Select Dir” command.

3) Select the “EXAMPLES” directory also using the “F3: Select Dir” command.

4) The programs in the “EXAMPLES” directory will be displayed in the Program Selections menu. Select the “M4020E1” program and press “F2: Edit Prog”.

5) SYSdev51 will be invoked and the main development menu will be displayed. Select “1: Edit program/On-line Funcs” and then the “F1: Main Prog” to view the program.

APPENDIX A
PROGRAMMING EXAMPLE

M4020 (PLC) program Example:
SYS51 System Configuration: M4020E1.LCF

System Configuration

Target Board:	M4020 8-Input/8-Output Intelligent PLS Module
Network Baud Rate:	344KBPS
Input0 Interrupt Enable:	No
Input1 Interrupt Enable:	No
Fixed Scan Time Mode:	No
Fixed Scan Time:	

APPENDIX A
PROGRAMMING EXAMPLE

SYS51 Main Program: M4020E1.LMN

The following is an example of an M4020 (PLC) program. The program provides various examples of ladder rungs, high-level instructions, and communications on the serial network, plus some of the documentation capabilities of SYSdev (including inter-block comments and variable annotation).

****

block: 1 - Ladder

The following block generates both a leading edge single shot (F000) and a trailing edge single shot (F001) on the timing channel 00 input from the PLS section of the M4020. Flag F000 is "on" for one scan when CH00 goes from "off" to "on". Flag F001 is "on" for one scan when CH00 goes from "on" to "off".

```
    Timing  Timing                                                  Timing
     CH00    CH00                                                    CH00
     input    bit                                                   L.E.S.S
    X000.0   F002                                                    F000
 0:+--] [--+--]/[--+-------+-------+-------+-------+-------+-------+--( )--


    Timing  Timing                                                  Timing
     CH00    CH00                                                    CH00
     input    bit                                                   T.E.S.S
    X000.0   F002                                                    F001
 1:+--]/[--+--] [--+-------+-------+-------+-------+-------+-------+--( )--


    Timing                                                          Timing
     CH00                                                            CH00
     input                                                           bit
    X000.0                                                          F002
 2:+--] [--+-------+-------+-------+-------+-------+-------+-------+--( )--
```

APPENDIX A
PROGRAMMING EXAMPLE

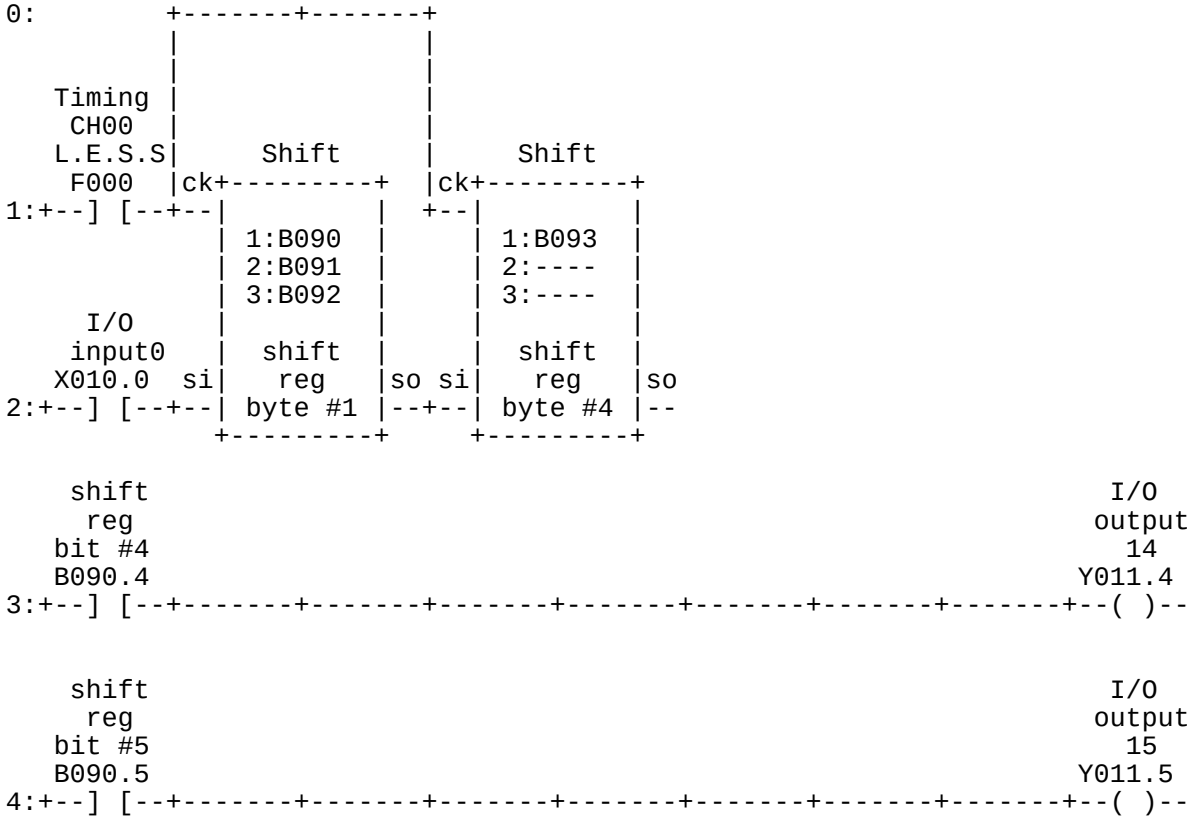
SYS51 Main Program: M4020E1.LMN

****

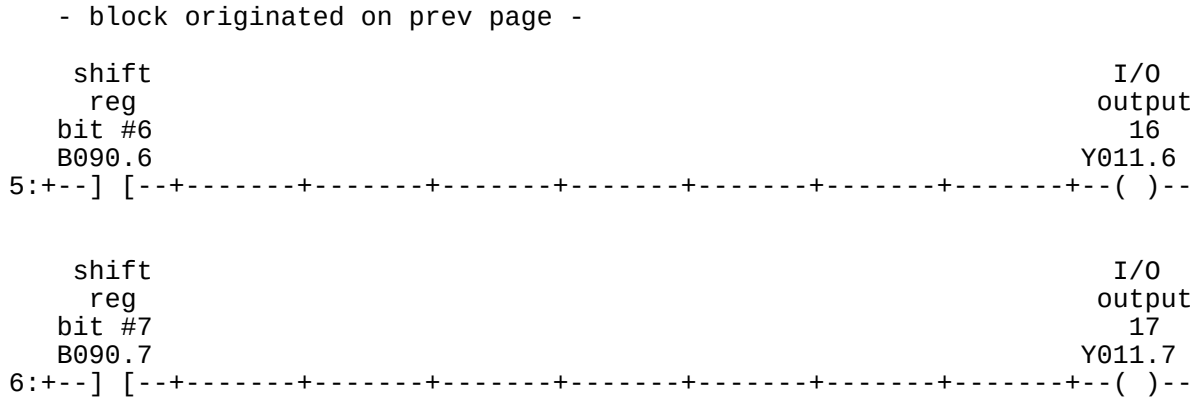
block: 2 - Ladder

The following block implements a 28 bit shift register. As can be seen, by cascading the shift register instructions, any number of shifts can be generated. Note that the input to the shift register "ck" is a leading edge single shot. The shift register shifts all the bits of the register left one bit every scan that the "ck" input is a "1". Thus if a leading edge single shot was not used, the shift register would be clocked every scan that "ck" input was a "1" which is generally not what is desired.

Each byte of the shift register is worth 7 shifts. Any bit within the shift register can be referenced in the program, as was done in the rungs following the shift register instructions.



- block continued on next page -


- block originated on prev page -
shift reg bit #6 B090.6
I/O output 16 Y011.6
5:
shift reg bit #7 B090.7
I/O output 17 Y011.7
6:

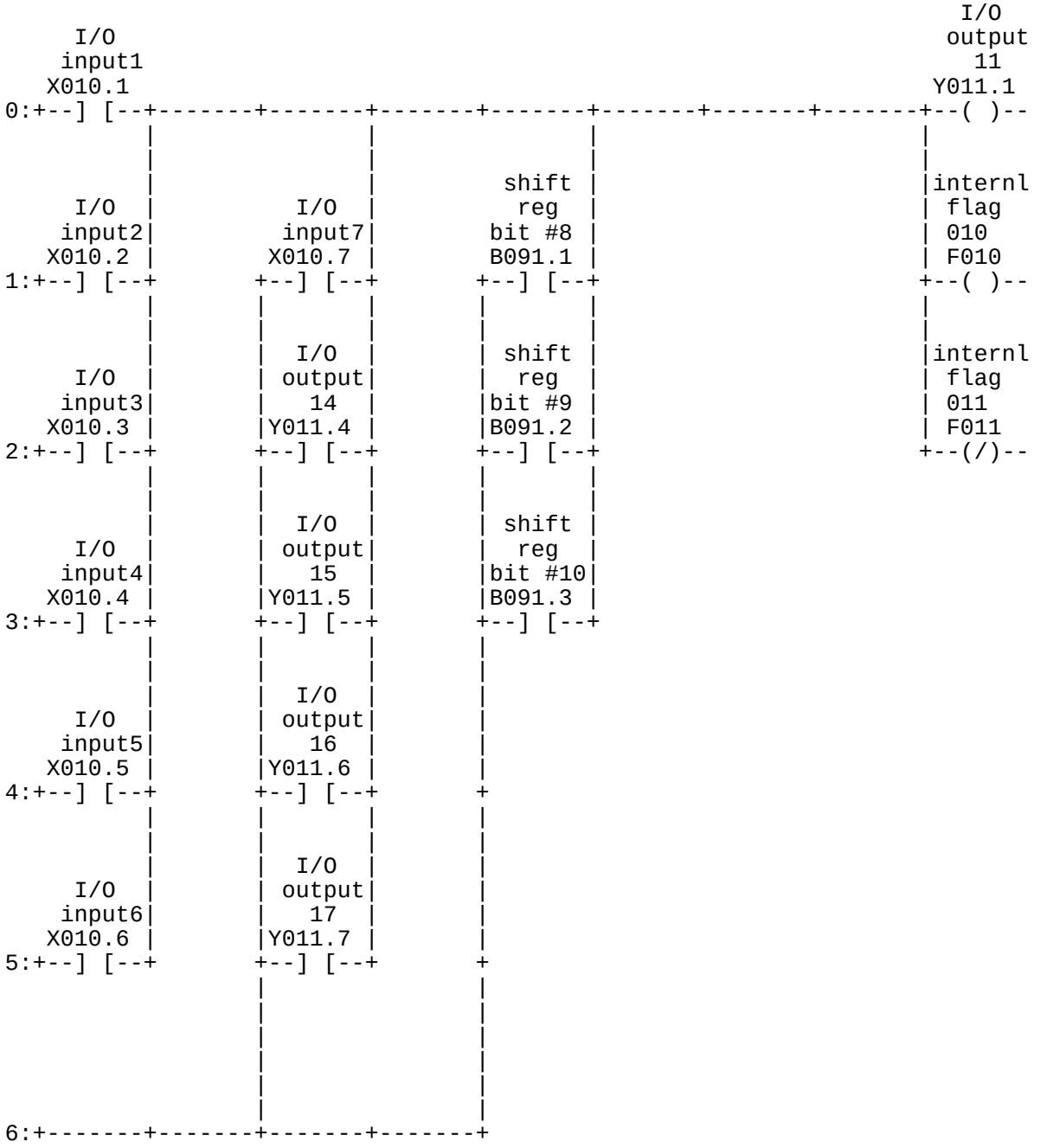
APPENDIX A
PROGRAMMING EXAMPLE

SYS51 Main Program: M4020E1.LMN

****

block: 3 - Ladder

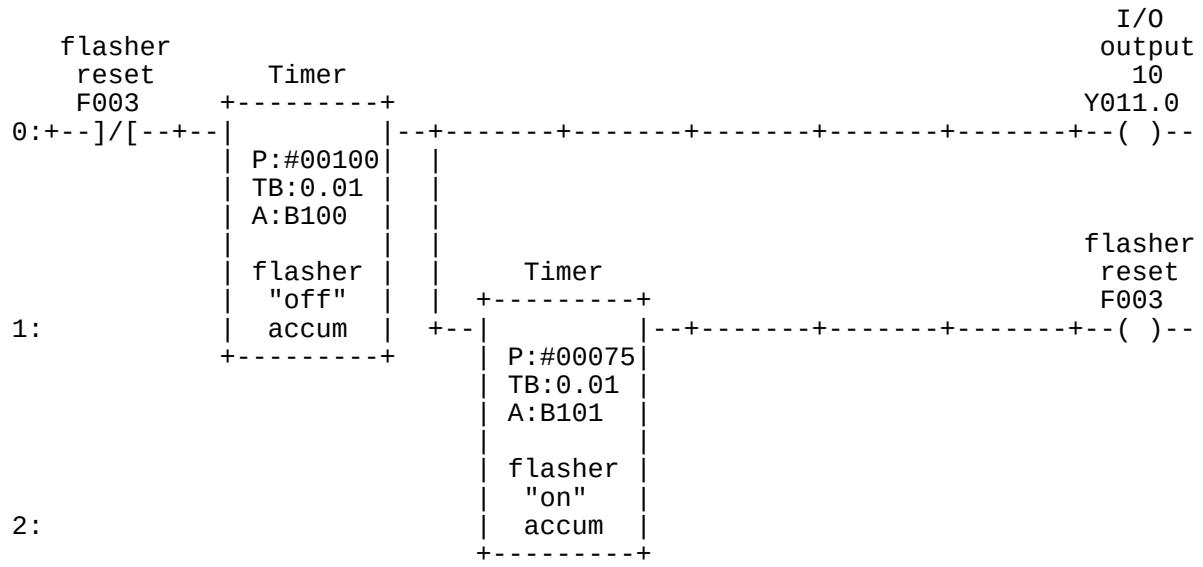
The following example shows 14 contacts "ored" together. This shows that even though the ladder block is 7 rows by 9 columns, large "or" statements can still be entered (up to 22 contacts ored together). Also note the coils "ored" together out the output of the rung. Up to 7 coils can be "ored" in one block.

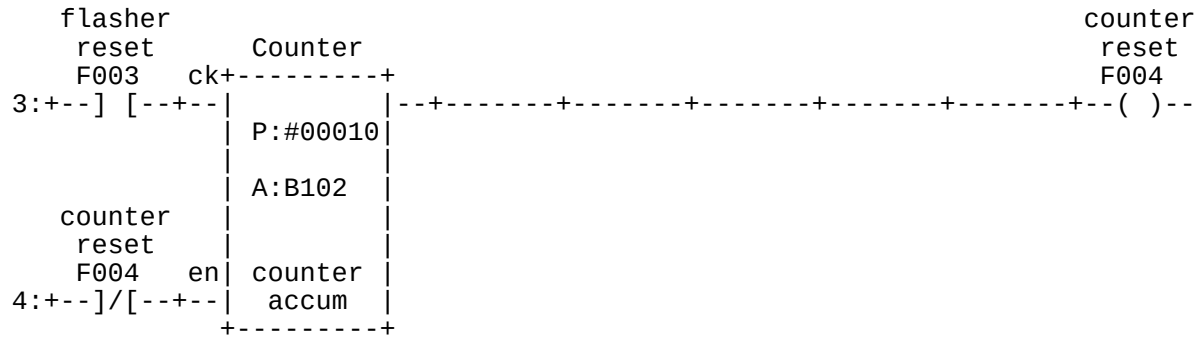


```
SYS51 Main Program: M4020E1.LMN

*************************************************************************
block:  4 - Ladder

This block implements a flasher circuit which flashes output Y11.0 "off"
for 1 second and "on" for .75 seconds. Note that F003 is only "on" for
one scan since the "off" timer is reset as soon as F003 is a "1" which
resets the "on" timer, turning F003 back "off" on the next scan. F003
is then used as the "ck" input to the counter in the next rung. Note
that the input to "ck" on the counter is a single shot, if this was not
the case the counter would down count once each scan that the "ck" input
is a "1", which is generally not what is desired. The counter resets itself
as soon as it down counts to 0 since F004 is set to a "1" which resets the
counter, setting F004 back to a "0" on the next scan.
```





APPENDIX A
PROGRAMMING EXAMPLE

```
SYS51 Main Program: M4020E1.LMN

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

The following block is an example of an "if else-if else" instruction
in a high-level block which simply compares the current value of the
counter accum from the previous block and sets Y11.2 and Y11.3 to
various states based on the current value. Also note the inter-block
comments in the high-level block with are separated with the "/*" and
"*/" comment delimiters. The compiler ignores all text after the "/*"
beginning delimiter until it detects the ending "*/" comment delimiter.
This allows any amount of comments to be entered in the block. Don't
forget the ending "*/" delimiter once you have entered the beginning
delimiter, or else subsequent high-level instructions will not be
compiled, being viewed simply as comments by the compiler.

 0:if (B102 >= 8)    /* counter accum greater than or equal to 8? */
 1:   {
 2:   Y11.2 = 1;     /* yes, set output 12 "on" and */
 3:   Y11.3 = 0;     /* output 13 "off" */
 4:   }
 5:else if (B102 >= 4 && B102 <= 7)     /* counter accum between 4 and 7? */
 6:   {
 7:   Y11.2 = 0;     /* yes, set output 12 "off" and */
 8:   Y11.3 = 1;     /* output 13 "on" */
 9:   }
10:else
11:   {              /* if counter accum is less than 4, */
12:   Y11.2 = 0;     /* set output 12 "off" and */
13:   Y11.3 = 0;     /* output 13 "off" */
14:   }
15:


B102  (-------)          counter  accum
Y011.2(-------)    I/O    output    12
Y011.3(-------)    I/O    output    13
```

```
SYS51 Main Program: M4020E1.LMN

************************************************************************
block:  6 - High-level
```

The following block is an example of communications between a master M4000 module (the module this program is running in) and a slave module on the serial network (address 2). The slave module has no program code pertaining to this communications, it responds automatically to this communications request.

The communications is implemented as follows:

1) In this example, communications is enabled when B67 is set to any value other than 0 (B67 can be written to using the "Assign Value" selection of the on-line monitoring menu). If B67 is zero, no communications on the network is performed, if B67 is any value other than zero, network communications is performed continuously. Once enabled, the following steps are performed by the sfunc13.

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

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

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

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

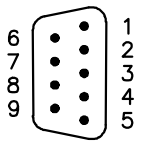
```
 0:if (B67 != 0)          /* enable serial comm? */
 1:   {
 2:   B65 = sfunc13(2,2,W80,W120,2,W110,W84);  /* yes, communicate with */
 3:   if (B65 != 1)                            /* slave #2.             */
 4:      B66 = B65;
 5:   if (B65 >= 3)       /* error code return value? */
 6:      B68 = B65;       /* yes, save error code in B68 */
 7:   }
 8:


B065  (-------)   comm   return   value
B066  (-------)   comm   return   value
B067  (-------)  serial   comm   enable
B068  (-------)   comm    error   code
W080  (-------)   master  send    stack
W082  (-------)  ------- ------- -------
W084  (-------)  master  receive  stack
W086  (-------)  ------- ------- -------
W110  (-------)   slave   send    stack
W120  (-------)   slave   receive  stack
```

APPENDIX A
PROGRAMMING EXAMPLE

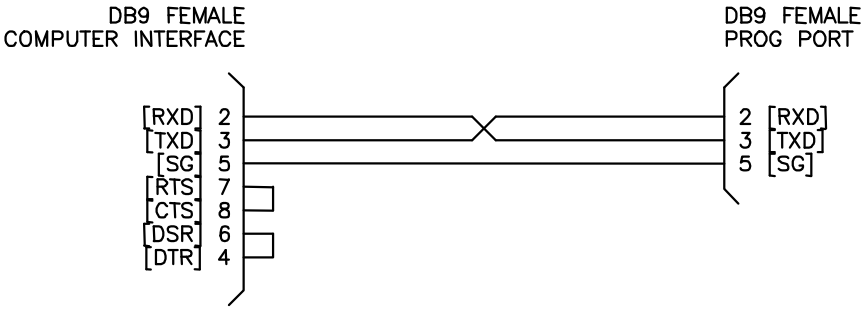
(This Page Intentionally Left Blank)

APPENDIX B
RS-232 PINOUTS/CABLES

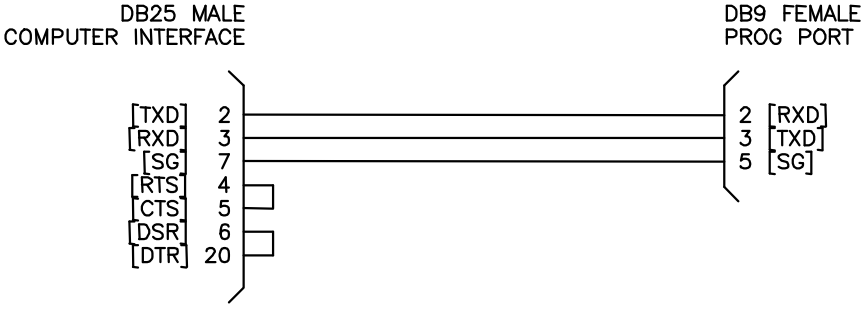


1: NOT USED
2: RXD (RECEIVE DATA – IN)
3: TXD (TRANSMIT DATA –OUT)
4: NOT USED
5: SG (SIGNAL GROUND)
6: NOT USED
7: NOT USED
8: NOT USED
9: NOT USED

PROG/CHAN Port Pin Out



DB9 (com1) to PROG Port Cable

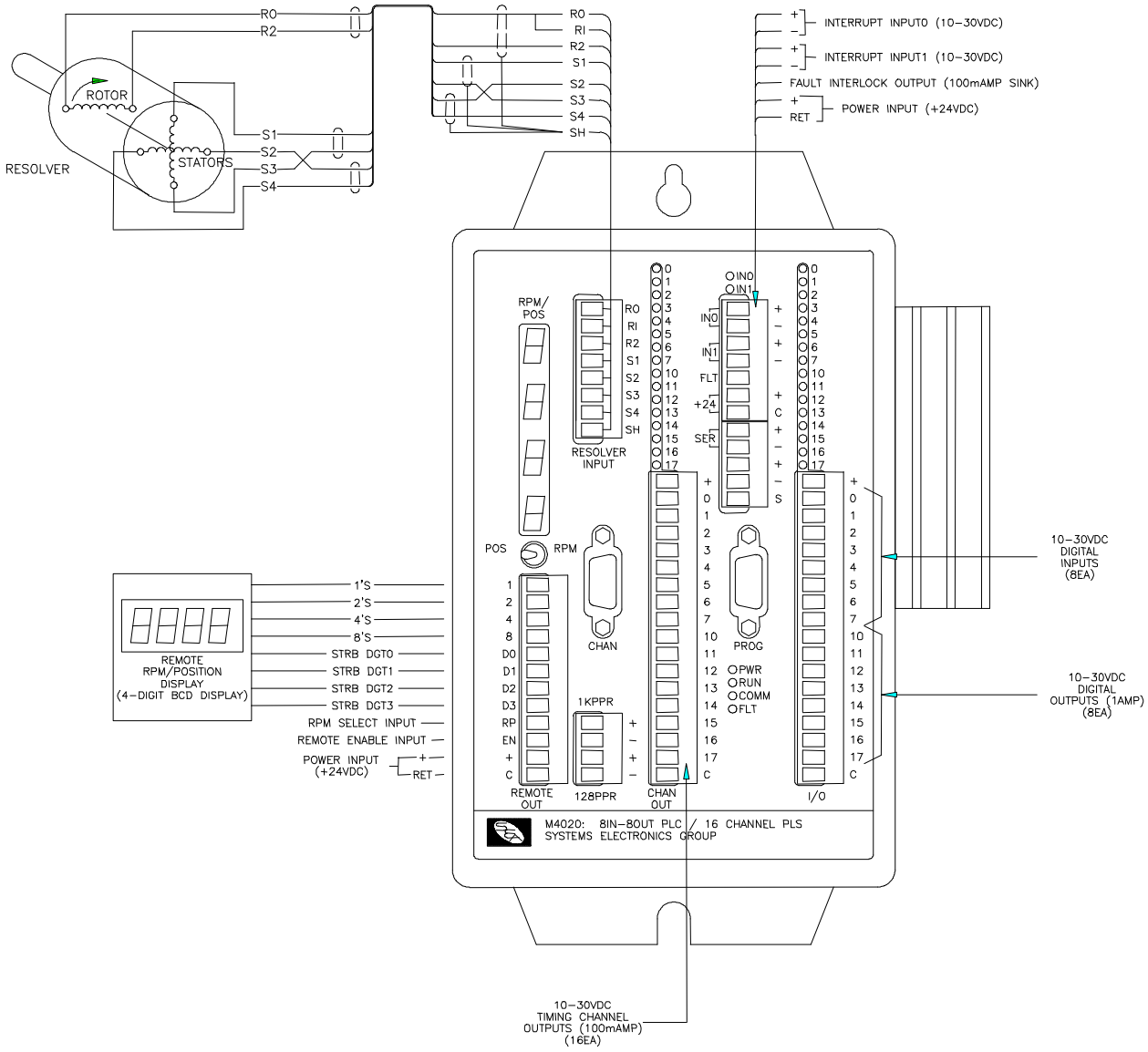


DB25 (com1) to PROG Port Cable

APPENDIX B
RS-232 PINOUTS/CABLES

(This Page Intentionally Left Blank)

APPENDIX C
FIELD WIRING CONNECTOR PIN-OUTS



M4020
8IN-8OUT PLC/16 CHANNEL PLS MODULE