MCS-920 Series

DC MOTION CONTROL SYSTEM

Installation & Operation Manual
MCS-921c

19 Linden Park

Rochester, NY 14625

July 21, 1989

TABLE OF CONTENTS

GENERAL DESCRIPTION

This manual covers the MCS-920 Series Motion Control Systems. The MCS-920 Series are designed to control DC Servomotors and include:

- all necessary power supplies for operation on standard 115 VAC
- a Programmable Motion Controller (PMC-904),
- a Pulse Width Modulated DC servodrive (300 watts to 1.8 kW output), and
- a mounting rack for 16 discrete I/O points (Opto-22 type).

The MCS-920 Motion Controllers control a wide range of DC brush-type Servomotors. Nine standard models are currently available, which are matched in power output to ORMEC's MTE Series Motors. These motors range from 3.3 lb-in to 48 lb-in continuous stall torque.

The MCS-920 Motion Controllers and ORMEC's MTE-Series DC Servomotors provide the following features:

Programmable Motion Control

ORMEC's PMC-904 with MPL+MATH high level Motion Programming Language provides the most powerful and easy to use Programmable Motion Controller available. It enables a motion control system designer to quickly and easily program a sophisticated motion control application. The I/O capability of the language, coupled with the arithmetic and logical functions, allow the motion controller to be programmed to communicate with the operator in the language of your specific industry.

MCS-920 Series Performance Features

- wide range of PWM output power (300 watts to 1.8 kilowatts)
- high peak currents (torques), which are typically double the rated current for up to a second
- internal peak and RMS current limits
- wide current, velocity and position loop bandwidths for high positioning accuracy and response
- all loop gains and compensators are implemented by software controlled hardware; This provides the ultimate in response but requiring no potentiometers for tuning adjustments.
- high quality analog tachometer signal is used for the hardware velocity loop, with the flexibility of software controlled parameters

Modular Construction

- all necessary power supplies are provided, including a line filter for control power and the optional 24 VDC I/O power supply
- integral circuit breakers for computer circuitry and servodrive bus power

Safety and Diagnostic Features

- protection features include servo bus overvoltage, undervoltage, overcurrent, main circuit breaker trip, Programmable Motion Controller (PMC) power failure, PMC RAM failure, ROM failure or program memory failure
- diagnostics provided by one dual color LED on the Programmable Motion Controller, and one red LED on the servodrive module
- the drive enable interlock allows the user to directly disable the output transistors in the PWM servodrive

- the forward amd reverse torque enable interlocks allow overtravel limits to disable torque while allowing the motor to back off the limit

Electrically Isolated Machine I/O Interface

- 16 I/O points (12 programmable), complete with I/O Rack which is compatible with Opto-22 DC or AC modules
- Optional +24 VDC power supply provides isolated power for DC I/O Modules

Servomotor Performance Features

- wide range of torque, from 3.3 lb-in to 48 lb-in RMS, conservatively rated
- maximum speeds from 2,500 to 4,000 RPM
- high torque to inertia ratios of the motors provides more delivered power in high performance applications
- durable construction includes quality brushes and rugged bearings
- built in tach and position encoder, with up to 12,000 counts per revolution
- low torque and tach-voltage ripples

SPECIFICATIONS

2.1 GENERAL MCS-920 SERIES SPECIFICATIONS

Environment:
Operating Temperature: 0 to +50°C
Storage Temperature: -20 to +85°C
Operating and Storage Humidity: 0 to 90%, non-condensing

Panel Mounting:
Size: 19.4"h x 11.8"w x 9.5"d
Mounting Method: Vertical base mounting, four #10 screws
Bolt Pattern: rectangular pattern, 18 $^{7}/_{8}$"h x 9 $^{3}/_{4}$"w

Voltage Requirements:
Incoming Line Voltage: 115 VAC, single phase, ±10%, 50/60 Hz
Minimum & Maximum Line Voltages: 103 VAC min & 127 VAC max

Incoming Power Requirement & Servodrive Bus Power

Motion Controller	Incoming Power Requirement		Servodrive Bus Power		Compatible
	Power	Current	Voltage	Current	Motor
MCS-920/A	0.6 kVA	5 amps	75	16/3.9	MTE-213
MCS-920/B	0.6 kVA	5 amps	75	16/4.6	MTE-262
MCS-920/C	0.6 kVA	5 amps	75	16/5.2	MTE-263
MCS-920/D	0.7 kVA	6 amps	75	16/5.3	MTE-350
MCS-920/E	0.8 kVA	7 amps	75	16/5.7	MTE-351
MCS-920/F	0.9 kVA	8 amps	100	16/5.4	MTE-352
MCS-920/G	1.3 kVA	11 amps	125	16/8.5	MTE-533
MCS-920/H	1.5 kVA	13 amps	125	26/12	MTE-535
MCS-920/J	1.8 kVA	15 amps	125	30/15	MTE-537

Other Servodrive Specifications:
Rated Servodrive Bus Voltage: 75, 100, 125 or 150 VDC
Corresponding Transformer Tap: 57, 74, 91 or 109 VAC
Bus Shunt Regulator Activation Voltage: 185 VDC
Excess Bus Voltage Fault: 203 VDC
Bus Undervoltage Fault: 45 VDC
Switching Frequency: 16 kHz for MCS-920/A - MCS-920/G
5 kHz for MCS-920/H & MCS-920/J

Encoder Power Voltage: 5 VDC ±5% @ 750 mA max

Input Interlock Circuitry:
Drive Enable (D-Enable) Input: contact closure (+12 VDC) to common to disable[1]

Overtravel Limit Switch Inputs:
Forward Enable (F-Enable) & Reverse Enable (R-Enable) Inputs: contact closure (+12 VDC) to common to disable

Power Supplies Available to User: +5 VDC @ 200 mA, ±12 VDC @ 200 mA
+24 VDC @ 1.2 A (optional)

---

[1] A "normally closed - held open" contact closure from the Servo Enable Relay (K2) in the PMC-100 is connected from this point to signal common. The user may also provide a parallel contact closure to disable the servodrive externally.

2.2 PROGRAMMABLE MOTION CONTROLLER (PMC-904)

CPU: type CMOS 8085A; speed 3.072 MHz

Power: +5 VDC ±5%, ±12 VDC, ±5%

Motion Control Program Storage: 8k bytes non-volatile RAM, lithium battery internal to the RAM chip, expected life of 20 years, used for program and register storage

Feedback:

Position Resolution: Dual channel quadrature with "square wave" outputs and an optional reference pulse, with resolution of "four times linecount"

Max Position Data Rate: 384 kHz; This rate refers to the decoded pulse train at a resolution of four times "linecount"; for this rate the "A" and "B" quadrature encoder channels are at 96 kHz.

Encoder Inputs: *and Machine Sensor (SENSIN)* differential, ±12 V max, 2 V threshold, changeable to 0 V, 40k ohm minimum input impedance, 4-32 u-sec scan time

Velocity: *LVTACH* analog, 0-60 VDC, 68k ohms input impedance
HVTACH 0-150 VDC, 338k ohms input impedance

Position Loop:

Maximum position error: ±2047 counts

Lag compensator break frequencies:

velocity loop	1.5 to 50 Hz
position loop	0.5 to 25 Hz

Power Requirements: +5 VDC, 1 A max
±12 VDC, 0.15 A max

Serial Communications:

Standards: EIA RS-232C, RS-422/485

Data bits: 8[2]

Start bits: 1 *Stop bits:* 1 *Parity:* none

Baud Rate: 38.4k 19.2k 9600 4800 2400 1200 600 300

Configuration:

The factory installed communications configuration is for RS-232C Data Communications Equipment (DCE) with hardware flow control. It is pin compatible with the serial port of an IBM-PC or compatible. Most RS-232 ASCII terminals will also communicate with this interface, however some do not provide the hardware flow control feature. Other options, such as RS-422/485 or RS-232 without flow control, are detailed in the Installation Section under PMC Configuration Jumpers.

---

[2] The PMC uses 8 data bits for communications, however normal ASCII terminal communications require only 7 data bits. The eighth (top) data bit is used to indicate that the servodrive is enabled in the } "prompt" character and for binary communications with host computers.

INSTALLATION

3.1 RECEIVING AND INSPECTION

The MCS Series Motion Controllers and their associated servomotors are put through severe tests at the factory before shipment. After unpacking, however, check to ascertain that they have sustained no damage while in transit. The motor output shaft should rotate freely by hand, and the bolts and screws should all be tight. Check the motion controller for any bent or broken components or other physical damage before installing.

Before mounting the servomotor: Wash off any anticorrosive paint which may be on the motor shaft and/or flange surface with paint thinner before attaching the motor to the driven machine. Oil, or otherwise treat for corrosion, the flange surface and shaft immediately after removing the paint. Also, during the test run, the driven machine should not be attached to the servomotor.

3.2 MOTION CONTROLLER INSTALLATION

3.2.1 Panel Mounting & Environment

Panel mounting information is available in the Specifications Section and Appendix B. The motion controller environment should be maintained as described below.

- Keep the temperature of the Motion Controller at 50°C or below.
- If the electrical panel is subjected to vibration, mount the unit on shock absorbing material.
- Avoid locations where corrosive gases exist as it may cause damage over long use. The switching contacts of contactors and relays are especially vulnerable.
- Select a location with minimum exposure to oil, water, hot air, high humidity, excessive dust or metallic particles.
- The preferred mounting orientation for the motion controller is vertical on a panel using the mounting holes (4) on the base plate, however it can also be mounted horizontally on its base.

3.2.2 MCS-920 Series Functional Wiring

The Functional Wiring Diagram which encompasses recommended safety and fault interlocks for a typical system is shown in Appendix A. **Before applying power, refer to Test Run Preparation in Section 5.**

The Functional Wiring Diagram also illustrates the connection of the servomotor, tachometer and position encoder, overtravel limits and Machine I/O, as well as the interconnection between the Programmable Motion Controller (PMC) and the servo-drive. A detailed description of the interconnections to the MCS-920 Series Motion Controllers is found in the following sections.

3.2.3 PMC-100 Series Internal Wiring

The MCS-920 is modularly constructed, with the PMC-904, I/O Rack, and the low voltage power supplies housed in a module called the PMC-100. A schematic of the interconnections to the PMC-100 is provided in Appendix E. The internal wiring of this unit is illustrated in Appendix F.

3.2.4 Recommended Motor & Motion Controller Wiring Methods

- When the motor is mounted to the machine and grounded through the machine frame, $^{dv}/_{dt}$ current flows from the PWM power through the floating capacity of the motor. To prevent noise effects from this current, and also for safety, the motor case should be connected to the Frame Ground (FG) of the MCS-920 (terminal 10 of JF2), which is factory connected directly to the MCS frame. The user should ground the frame of the MCS-920.
- To prevent noise on the encoder signals, shielding should be used which can be terminated at terminal 5 of JF2.
- When motor wiring is contained in metal conduits, the conduits and boxes should also be grounded. Use multi-strand wires of 12 AWG or heavier for grounding to the case (preferably flat woven silver plated copper braid).
- Route signal and power leads in separate conduit or ductwork.

3.3 INPUT POWER - TERMINAL BLOCK TB3

The MCS-920 Series Motion Controllers operate on standard, unisolated 115 VAC power. This power is used for both the motion control electronics and the main servodrive bus power supply (electromotive power). **Since, in normal operation, a servomotor is "ON" even when stopped, it can present a hazardous situation to personnel in certain applications. In those cases, it is strongly recommended that the user interlock the main power of the MCS with suitable Emergency Stop and/or safety guard circuitry.**

The incoming power requirements are detailed in the Specifications Section. Terminal block TB3 is provided for connecting power to the unit. Note that the MCS-920 (PMC-100) includes two circuit breakers and a switch for power. The 2.5 amp breaker, for the low voltage power supply is interlocked with the switch, insuring that loss of Motion Controller Control Power will disable Servodrive Control Power. The 15-amp breaker is for the Servodrive Main Bus Power Supply. There is an integral line filter to provide industrially hardened low voltage power.

Terminals are clearly marked as indicated below:

Symbol	Name	Description
L N	Input Power	Single-phase 115 VAC, ±10%, 50/60 Hz, with a switch (top of unit) and built-in circuit breakers.
FG	Frame Ground	Connect to the enclosure chassis and earth ground, preferably with AWG 8 or larger braid.
B N	A.C.Servodrive Bus Power	Single-phase 115 VAC power distribution for the servodrive main bus power and fan is interlocked with the PMC power switch and the 20 amp circuit breaker inside the PMC-100 chassis. These terminals are factory wired to provide the transformer and fan power.
N T	Transformer Alarm Contact	These terminals attach to a normally closed contact in the servodrive bus power supply transformer. Should a thermal overload be detected, the contact will open, dropping out the servo enable relay (K2).

3.4 MOTOR-TACH-ENCODER - CONNECTOR JF2

Connector JF2 is provided for connecting the Motor-Tach-Encoder. Terminals are assigned as indicated below, where their individual function is briefly described.

Signal	Location	Function	Description
TACH	1	DC Tachometer	The DC Tachometer input provides an analog signal proportional to the velocity of the servomotor. This signal is used by the Programmable Motion Controller to close an analog velocity control loop in hardware, which enhances the system stiffness and high frequency performance. For ORMEC's MTE series servomotors, the TACH signal has an output of 7 volts/kRPM.
TACH COMM	2	DC Tachometer Common	This input provides the connection to signal common for the DC Tachometer.
+5 VDC	3	Encoder Power	This output pin provides +5VDC power for use by the incremental position encoder.
COMM	4	Encoder Power Common	This output pin provides the connection to signal common for the position encoder.
ENCA	7	Encoder Channel A	This input is one of two quadrature square wave signals from the position feedback encoder.
ENCA'	8	Encoder Channel A	ENCA' is the logical complement of ENCA, and is present when the position encoder used has differential outputs, such as in ORMEC's MTE Series Servomotors.
ENCB	11	Encoder Channel B	This input is one of two quadrature square wave signals from the position feedback encoder.
ENCB'	12	Encoder Channel B	ENCB' is the logical complement of ENCB, and is present when the position encoder used has differential outputs, such as in ORMEC's MTE Series Servomotors.
ENCR	9	Encoder Channel R	This input is the incremental position encoder "reference" or "marker" channel. In the standard MCS-920 Series configuration, Encoder Channel R is attached to the "once per revolution" reference signal derived from the encoder mounted on the servomotor.
ENCR'	13	Encoder Channel R	ENCR' is the logical complement of ENCR, and is present when the incremental position encoder used has differential outputs, such as in ORMEC's MTE Series Servomotors.
SHIELD	5	Shield Ground	This connector pin is for terminating the shields of the Motor-Tach-Encoder Cable to reduce electrical noise.
MOTOR 2	14	Motor Armature	This output pin connects to one side of the DC Servomotor armature.
MOTOR 1	6	Motor Armature	This output pin connects to one side of the DC Servomotor armature.
FG	10	Frame Ground	This pin is provided to ground the case of the servomotor for safety considerations.

3.5 SERVODRIVE INPUT CONNECTIONS - TERMINAL BLOCK TB5

Terminal Block TB5 is provided for connecting the servodrive enable and the overtravel limit switches (if any). It also includes terminals for the ±12 VDC power supply and the optional +24 VDC I/O power supply for use with the Machine I/O. Two informational signals, SDRVEN' and CUR-MON are also available on this terminal block.

3.5.1 Servodrive Interlock Inputs

The servodrive interlock inputs are designed for compatibility with contact closures to ground (GND). During normal operation (and with no interlocks enabled), all three of the contact closures should be open.

Signal	Location	Function	Description
D-ENABLE	TB5-6	Drive Enable	Shorting this input to ground (TB5-8) will disable the output transistors of the servodrive, preventing armature current (and therefore torque) in the servomotor. It should be noted that this terminal is a "wired or" terminal and is driven by both the servodrive fault detection circuitry and a "normally closed" contact of relay K2 in the PMC-100. See the description below for Monitor Signal SDRVEN'(TB5-1). In addition, the user can disable the servodrive at any time by shorting D-ENABLE to ground.
F-ENABLE	TB5-5	Forward Enable	This input must not be shorted to ground for the servomotor to provide torque in the "forward" direction. For applications where it is desirable to disable forward torque, e.g. a ballscrew drive with overtravel limit switches, connect this input to ground (TB5-8) through a "normally open" switch. If the switch "closes", shorting this input to ground, forward torque will be disabled. The servodrive can still provide "reverse" current (torque) allowing the servomotor to "back away" from the limit.
R-ENABLE	TB5-4	Reverse Enable	This input must not be shorted to ground for the servomotor to provide torque in the "reverse" direction. For applications where it is desirable to disable reverse torque, e.g. a ballscrew drive with overtravel limit switches, connect this input to ground (TB5-8) through a "normally open" switch. If the switch "closes", shorting this input to ground, reverse torque will be disabled. The servodrive can still provide "forward" current (torque) allowing the servomotor to "back away" from the limit.
D-RESET'	TB5-3	Servodrive Reset	Shorting this input to ground (TB5-8) resets the DRIVE FAULT circuitry of the servodrive, turning off the DRIVE FAULT LED indicator and restoring normal operation. A DRIVE FAULT occurs when the protective circuitry of the servodrive detects a fault condition and turns off the output transistors preventing armature current (and therefore torque) in the servomotor. The SERVO DRIVE RESET pushbutton switch on the MCS shorts this input to ground when pressed.

3.5.2 Available Power Supplies

+12 VDC	TB5-7	+12 volts DC	This power supply is used for RS-232 communications and analog circuitry and on the PMC and the servodrive, and is available for the user to power I/O circuitry, sensors or other auxiliary devices. Since it is used by both the PMC and the servodrive, take care in loading and shielding if it is used for other purposes. The available current can be found in the Specification Section.
-12 VDC	TB5-9	-12 volts DC	This power supply is used for RS-232 communications and analog circuitry on the PMC and the servodrive, and is available for the user to power I/O circuitry, sensors or other auxiliary devices. Since it is used by both the PMC and the servodrive, take care in loading and shielding if it is used for other purposes. The available current can be found in the Specification Section.
GND	TB5-8	±12 VDC Common	This terminal is the common for the ±12 VDC power supply, as well as the PMC and servodrive circuitry in the MCS-920.
+24 VDC	TB5-11	I/O Power	The optional fully isolated +24 volt DC power supply is available for use with DC Machine I/O modules. The available current can be found in the Specification Section.
+24 COM	TB5-10	I/O Power Common	Common for the optional +24 volt DC power supply; This power supply common is floating, and can be grounded by the user as desired.

3.5.3 Monitor Signals

The following monitor signals are available to determine whether or not the PMC-904 Programmable Motion Controller is attempting to enable the servodrive, and the amount of armature current being output by the servodrive.

SDRVEN'	TB5-1	Servodrive Enable'	This TTL level signal is provided by the PMC to enable or disable the servodrive. It drives the Solid State Relay (K1), which operates the Servo Enable Relay (K2). The isolated output contact of this relay drives D-ENABLE (TB5-6) to enable and disable the servodrive. If either 115 VAC power or +5 VDC power is lost to the PMC, then the "normally closed" contact of relay K2 will short D-ENABLE to ground, disabling the servodrive.
CUR-MON	TB5-2	Current Monitor	This output current monitor signal from the servodrive is capable of driving a zero center 3 milliampere ammeter. The scale factor is as follows: MCS-920/A - MCS-920/G: 1.6 amps/volt MCS-920/H - MCS-902/J: 3.0 amps/volt

3.6 MACHINE I/O CONNECTIONS - TERMINAL BLOCK TB4

The Machine I/O Interface is provided to allow the MCS to interface with the machine or other systems using 16 discrete digital I/O points. This interface is an Opto-22 type 16 position I/O rack which is compatible with a variety of interface modules, and is wired at TB4. The I/O points are functionally labelled, and both the labels and the corresponding terminal numbers are shown in Appendix A.

The interface consists of 11 inputs and 5 outputs. Standard uses of Machine I/O include:

Signal	Location	Function	Description
OUT1	TB4-1,2	Discrete Output	These four discrete outputs are software controlled. When asserted, the PMC TTL output will be low (approximately 0 volts), causing the indicator LED to be "ON" (only when Opto-module installed) and the respective Opto-module to conduct output current.
OUT2	TB4-3,4	Discrete Output	
OUT4	TB4-5,6	Discrete Output *Fault Output*	Note that OUT4 and OUT8 can be configured to operate as Fault and In Motion outputs, respectively. See Section 4.6.2.
OUT8	TB4-7,8	Discrete Output *In Motion Output*	
IN1	TB4-9,10	Discrete Input	These eight discrete inputs are accessible by a number of MPL software commands, to control program flow or read in data. When voltage is applied to the Opto-module, the indicator LEDs will be "ON" and the PMC inputs asserted (TTL low).
IN2	TB4-11,12	Discrete Input	
IN4	TB4-13,14	Discrete Input *- Overtravel Limit*	For applications such as ballscrew drives with overtravel limit switches, IN4 and IN8 can be set up as inputs for the switches. See Section 4.6.1.
IN8	TB4-15,16	Discrete Input *+ Overtravel Limit*	
IN10	TB4-17,18	Discrete Input *Address Line-1*	For applications where the user would like to access more than two MPL programs from Discrete I/O, some of these inputs can be used to provide a "5 line address" for the routine to be selected. This feature allows up to 32 MPL programs to be selected from Discrete I/O. See Section 4.6.3.
IN20	TB4-18,20	Discrete Input *Address Line-2*	
IN40	TB4-21,22	Discrete Input *Address Line-3*	
IN80	TB4-23,24	Discrete Input *Address Line-4*	
SEL	TB4-25,26	Program Select *Address Line-5*	The SEL input wil select one of two programs when the READY output is ON and the EXECUTE input is asserted. The MPL routine **@0** will be run if SEL is asserted (LED ON), otherwise **@1** will be run.
EXECUTE	TB4-27,28	Program Execute	
READY	TB4-29,30	MPL Ready	The READY output is asserted whenever the PMC is ready to accept a command or run a new routine.
STOP	TB4-31,32	Stop Program (& Motion)	Asserting STOP will stop motion and "break" (stop the execution of) any MPL program currently running. If this input is asserted when the unit is powered up or reset, the automatic powerup sequence is shortened. See the Powerup discussion in the Operation Section.

3.7 PMC MOTOR LOOP INTERFACE - TERMINAL BLOCK TB1 (TM1)

The PMC motor loop interface is provided by a "pluggable" terminal block, which plugs into mating connector TM1 on the PMC. It includes connections to disable the servodrive output current, as well as provide velocity and position feedback to the PMC. In addition it provides an interface for a fast machine sensor, such as a registration mark detector, as well as a velocity command input and output for use in Electronic Lineshaft Applications.

3.7.1 Interconnections between the PMC and the Servodrive

Signal	Location	Function	Description
ICMD	TB1-1	Current Command	This analog output of the PMC is used to command servodrive output current. It is scaled so that ±10 VDC will command approximately ± full current in the servomotor (and therefore full torque).
COMMON	TB1-2	ICMD Common	Signal common for the current command output.
SDRVEN	TB1-3	Servodrive Enable	This TTL level digital output signal is "high" (+4 to +5 VDC) when the PMC enables servodrive output power. It is provided for information only in the MCS-920 Series.
SDRVEN'	TB1-4	Servodrive Enable	This TTL level digital output signal is "low" (0 to 1 VDC) when the PMC enables servodrive output power. In the MCS-920 Series, it controls internal relays which provide a "normally closed - held open" contact output to drive the servodrive's (D-ENABLE) input.
SHIELD	TB1-5	Frame Ground	The SHIELD input is an interconnection point for cable shields provided to reduce signal noise. Internal to the PMC-100, this point is connected to the chassis frame.
POLARIZE	TB1-6	Polarization Plug	This pin is omitted from TM1 and plugged in TB1 to polarize the terminal block connection.
HVTACH	TB1-7	Velocity Feedback	This analog input is for DC analog velocity feedback when the maximum anticipated voltage is between 60 and 150 VDC. It is unused in the MCS-920 Series as shipped from the factory.
LVTACH	TB1-8	Velocity Feedback	This analog input is used for the DC analog velocity feedback when the maximum anticipated voltage is less than 60 VDC. It is connected to the TACH connection of JF2 in the MCS-920 as shipped from the factory. ORMEC's MTE-210, 260, 350 and 530 Series Motor-Tach-Encoders have tach gains of 7 volts/kRPM, which are within this standard range.
COMMON	TB1-9	Tach Common	Signal common for the Velocity Feedback input.

3.7.2 Position Feedback to the PMC

The factory standard configuration of the MCS-920 Series is to use position feedback derived from the integral position encoder built into the servomotor. These signals are input to the MCS-920 Series through connector JF2.

Many applications benefit from using position feedback attached directly to the load in lieu of the position encoder mounted at the rear of the servomotor.

Examples of this include the use of linear scales or encoders on the output of a gearbox. This is readily accomplished in the MCS-920 Series by disconnecting the encoder signals on either TB1 or JF2, and connecting the quadrature signals from the desired position feedback transducer. Normally, the analog velocity signal provided by the tachometer on the servomotor is still used, which allows a high performance velocity loop directly around the servomotor, enhancing performance in this "distributed feedback" situation.

The position feedback connections to the PMC are outlined below:

Signal	Location	Function	Description
+5 VDC	TB1-10	Encoder Power	The +5 VDC logic power supply used by the PMC is provided at this terminal for powering the incremental position encoder.
ENCA	TB1-11	Encoder Channel A	This input is one of two quadrature square wave signals used as feedback by the PMC.[3] It is a digital input signal with a "low" level of 0 VDC and a "high" level of between +5 VDC and +24 VDC.
ENCA'	TB1-12	Encoder Channel A	ENCA' is the logical complement of ENCA, and is present when the incremental position encoder has differential outputs, such as ORMEC's MTE-Series DC Servomotors. If single ended encoders are used this signal may be either left open or grounded.
ENCB	TB1-13	Encoder Channel B	This input is one of two quadrature square wave signals used as feedback by the PMC.[3] It is a digital input signal with a "low" level of 0 VDC and a "high" level of between +5 VDC and +24 VDC.
ENCB'	TB1-14	Encoder Channel B	ENCB' is the logical complement of ENCB, and is present when the incremental position encoder has differential outputs, such as ORMEC's MTE-Series DC Servomotors. If single ended encoders are used this signal may be either left open or grounded.
ENCR	TB1-15	Encoder Channel R	This input is the incremental position encoder "reference" or "marker" channel.[3] In the standard MCS-920 Series configuration Encoder Channel R is attached to the "once per revolution" reference signal derived from the encoder mounted on the servomotor. It is a digital input signal with a "low" level of 0 VDC and a "high" level of between +5 VDC and +24 VDC.
ENCR'	TB1-16	Encoder Channel R	ENCR' is the logical complement of ENCR, and is present only when the incremental position encoder used has differential outputs, such as the standard S-Series configuration. If single ended encoders are used this signal may be either left open or grounded.
SHIELD	TB1-17	Frame Ground	This terminal point is for cable shields used to reduce signal noise. In the standard MCS-920 Series, it is used to terminate the shield for the cable which connects to the servodrive, Connector CN1. Internal to the MCS-920 Series, it is connected to the chassis frame.

[3] The interfaces to the encoder and the machine sensor are configured for either single ended or differential inputs of up to 24 VDC, with a nominal switching point of 2 VDC. The switching threshold is controlled by the value of SIP Resistor network RN3 on the PMC, which is socketed for possible field change. This input circuitry is described in Appendix M.

3.7.3 Miscellaneous Connections to the PMC

Signal	Location	Function	Description
SENSIN	TB1-18	Machine Sensor	The Machine Sensor input may be used to synchronize motion to an external event.[3] Motion can be primed to start, begin deceleration or stop on this signal, which is "scanned" on each count of the velocity reference signal. In the 384 kHz velocity range the "scan time" is less than 3 microseconds.
SENSIN'	TB1-19	Machine Sensor	SENSIN' is the logical complement of SENSIN, and is present only when the machine sensor used has differential outputs. If a single ended sensor is used this signal may be left open or grounded.

Signal	Location	Function	Description
EXVCMD	TB1-20	External Velocity Command	The External Velocity Command input is an analog input to the analog velocity summing junction of the PMC. As shown in Appendix J-4, it goes to Pin 4 of Header J7. Default strapping as defined in the PMC Configuration Jumper Section of this manual routes this input (via Pin 3 of Header J7) to the velocity summing junction. Moving this jumper to short from Pin 4 to Pin 2 on Header J7, routes the input through an adjustable gain stage (FFGAIN) shown in Appendix J-2. **Note that when it goes through the adjustable gain stage, this signal must have a positive polarity, and a maximum voltage of +2.5 VDC.**
XVELREF	TB1-21	External Velocity Reference	The External Velocity Reference output is a bipolar analog output which is proportional to the commanded velocity of the PMC. With the factory standard PMC Configuration Jumper strapping of pins 6-8 on Header J7, this signal is positive for forward motion and negative for reverse motion. The polarity can be reversed by either moving the jumper from pins 6-8 to pins 7-8, or by setting the direction invert option in the X-Register. The gain of this signal is software selectable using the **TX** command, with a gain of 255 setting approximately 10 volts for the maximum speed of the current Velocity Range. The maximum output current is 5 milliamps.

3.8 SYSTEM AXIS INTERFACE - TERMINAL BLOCK TB2 (TM2)

The PMC system axis interface is provided by a "pluggable" terminal block, which plugs into mating connector TM2 on the PMC. The system axis interface provides for connection of the power supply voltages as well as an external position reference (EXREF) used for electronically lineshafted systems, an input to reset the PMC microprocessor, and a stop input. TB2 is a removable Phoenix type MSTB 1.5/10ST terminal block (or equivalent) and one is included with each unit.

Signal	Location	Function	Description
EXREF	TB2-1	Motion Reference Bus	The EXREF I/O point is used as an external position reference for electronically lineshafted systems. It is RS-485 compatible, and interfaced to both receiver U28 (SN75174N or equivalent) and tri-state driver U29 (SN75174N or equivalent). MPL software commands determine whether the PMC is driving, receiving or ignoring this external reference signal. Minimum signal acceptance time is 1.3 us, with a maximum input frequency of 350 kHz. If this signal is to be used, it is recommended that it be "pulled down" to COMMON by a 1k to 10k ohm resistor at one of the PMCs.
EXREF'	TB2-2	Motion Reference Bus	EXREF' is the logical complement of EXREF. If this signal is to be used, it is recommended that it be "pulled up" to +5 VDC by a 1k to 10k ohm resistor at one of the PMCs.
COMMON	TB2-3	Power Supply Common	This terminal is the power supply common for the power supply powering the PMC.
+12 VDC	TB2-4	+12 VDC power	This terminal is the +12 VDC power supply used for the PMC's analog circuitry and RS-232 communications. Any user connections to this power supply should be made at terminal block TB5.
-12 VDC	TB2-5	-12 VDC power	This terminal is the -12 VDC power supply used for the PMC's analog circuitry and RS-232 communications. Any user connections to this power supply should be made at terminal block TB5.
+5 VDC	TB2-6	+5 VDC power	This terminal is the +5 VDC power supply used for the PMC's digital circuitry. Up to 500 mA of +5 VDC is available for the user.
SHIELD	TB2-7	Frame Ground	EMI isolated SHIELD is an interconnection point for cable shields provided to reduce signal noise. Internal to the MCS-920 Series, this point is connected to the chassis frame.
POLARIZE	TB2-8	Polarization Plug	This pin is omitted from TM2 and plugged in TB2 to polarize the terminal block connection.
STOP'	TB2-9	Stop/Break Input	Asserting this input (shorting it to ground) will STOP the current motion and BREAK the current MPL program. Scan time is 4 msec.
RESET'	TB2-10	CPU Reset	Shorting this input to ground will reset the PMC system, including the on board microprocessor. **WARNING: Since this signal is tied directly to the RESET circuitry of the microprocessor in the PMC, proper caution should be exercised in its use. e.g. Don't tie long wires to this signal and use appropriate shielding.**

3.9 SERIAL COMMUNICATIONS INTERFACE - CONNECTOR JM2

The serial communications interface is provided by an industry standard 25 pin male "D-Series" connector. With the factory default communications jumper strapping in place, this interface is pin compatible with an IBM-PC serial port. It is also configurable for RS-422/485 for Multi-Axis Serial Bus Communications. See the PMC Configuration Jumper section for a description of jumper strapping.

3.9.1 Serial Interface - RS-232

This IBM-PC compatible RS-232 serial interface consists of the following signals:

Signal	Location	Function	Description
TxD	JM2-2	Transmit Data	This input to the PMC is where data transmitted by the serial communications device is received.
RxD	JM2-3	Receive Data	This output from the PMC is where data to be received by the serial communications device is sent.
CTS	JM2-5	Clear to Send	This output from the PMC should be monitored by the serial communications device to know when it is OK to send serial data to the PMC.
RTS	JM2-4	Request to Send	This input to the PMC is monitored by the PMC to know when it is OK to send serial data to the serial communications device.
COMMON	JM2-7	Signal Common	This pin is the signal common for the RS-232 serial communications signals.

3.9.2 Multi-Axis Serial Bus Interface - RS-422/485

This multi-axis Serial Bus Interface consists of the following signals. Each pair of signals are the outputs of a balanced differential line driver which conforms with the RS-422/485 standard, such as the TI SN75174. They are received by balanced differential receivers such as the TI SN75175.

Signal	Location	Function	Description
RDA	4	Receive Data'	This input to the PMC is where data transmitted by
RDB	16	Receive Data	the serial communications device is received.
SDA	6	Transmit'	This output from the PMC is where data to be
SDB	18	Transmit	received by the serial communications device is sent.
CSA	7	Clear to Send'	This output from the PMC should be monitored by
CSB	19	Clear to Send	the serial communications device to know when it is OK to send serial data to the PMC.
RSA	9	Request to Send'	This input to the PMC is monitored by the PMC to
RSB	21	Request to Send	know when it is OK to send serial data to the serial communications device.
+5V	25	+5 VDC serial power	The PMC power supplies can be provided to power
+12V	11	+12 VDC serial power	user serial devices such as the ORMEC SBI Series
-12V	23	-12 VDC serial power	Serial Bus Interfaces or the ITM-270 Industrial
COMMON	1	serial power common	Numeric Keypad. See the PMC Configuration Jumpers section for details.

3.10 PMC CONFIGURATION JUMPERS

The PMC has five header connectors which provide user selectable hardware options by the placement of "push-on" jumpers. **See Appendix H for a pictorial of the PMC with the Factory Default Configuration jumpers, as well as the alternate RS-422 Serial Bus configuration.**

3.10.1 Assigning Axis-ID - Header J6

Axis-IDs are assigned to PMCs to allow multiple units to communicate with a host computer or programming terminal through a single serial communications interface. This communications is accomplished using a "serial bus" technique, which connects all the PMCs on a "party line".

To use a serial communications bus, the PMCs on it must each have a unique Axis-ID **and** they must have their Serial Communications Interface configured for RS-422/485. For systems with up to 14 PMCs on the bus, the Axis-ID is normally assigned by the placement of Jumpers on Header J6. For systems with 15-26 PMCs on the bus, the additional Axis-IDs must be assigned in non-volatile RAM. Serial bus communications operation is described in the Operation Section.

Hardware Axis-ID Selection

	Header J6 Jumper Straps				Axis-ID
	1-2	3-4	5-6	7-8	
					Serial Bus Disabled
	X				A
		X			B
	X	X			C
			X		D
	X		X		E
		X	X		F
	X	X	X		G
				X	H
	X			X	I
		X		X	J
	X	X		X	K
			X	X	L
	X		X	X	M
		X	X	X	N
DEFAULT	X	X	X	X	Assigned from non-volatile memory

X - Indicates Strap Present

3.10.2 Serial Communications - Header J5

The Serial Communications Interface can be readily configured for RS-232 or RS-422/485 using jumper straps on Header J5.

IBM-PC Compatible - RS-232 DCE (Default)

Default Jumper Strap Positions for Header J5					
7-8	9-10	15-16	17-18	19-21	29-31

This serial communications selection is suitable for:
- any programming device supplied by ORMEC
- IBM-PCs or compatibles using ORMEC's development software (MAX-II)
- most RS-232 terminals.

To disable the PMC's hardware handshake interface lines for communications to a "dumb" terminal, remove jumper 7-8 and 9-10 and reposition them to connect pins 6-8 and 10-12.

Multi-Axis Serial Bus - RS-422/485 DCE

Alternate Jumper Strap Positions for Header J5							
11-12	13-14	21-22	23-24	25-26	27-28	31-32	33-34

If an you are using an ORMEC Serial Bus Interface module (SBI) to convert RS-232 to RS-422 for serial bus communications, it is possible to supply power to the SBI from your PMC. Install **additional** jumpers as follow:

Additional Jumpers for Supplying Power to ORMEC SBI Modules

1-2 3-4 5-6 19-20

This serial communications selection is suitable for:
- ORMEC's SBI-910-RS or SBI-911-SS (either is recommended)
- ORMEC's SBI-900-RC or SBI-911-SC (with standard configuration)
- other RS-422 communications devices which need power.

3.10.3 Feedforward / External Velocity Reference - Header J7

Default Jumper Strap Positions for Header J7		
1-2	3-4	6-8

Jumpers 1-2 and 3-4 may be removed, and a jumper placed from 2-4 to allow an external signal to provide the feedforward reference. The external signal should have a range of 0 to +2.5 volts maximum. This option allows the system to accurately track an externally provided motion reference in the "motion bus slave" mode.

The jumper from 6-8 may be removed and placed from pin 7-8 to change the polarity of the external velocity reference signal.

3.10.4 Daughter Board Connector - Header JM4

Default Jumper Strap Positions for Header JM4
25-26

The jumper strap above must be in place unless there is a daughter board added to the PMC on Header JM4.

3.10.5 Motion Reference Bus Selection - Header JM20

Default Jumper Strap Positions for Header JM20	
1-2	4-5

The jumper strap from pin 1-2 connects the Motion Reference Bus receiver ouput to the internal command pulse generation circuitry of the PMC. In Electronic Lineshafting applications where an EBC-900 daughter board is used to interface the PMC to an encoder which is providing remote motion reference data, then it must be removed. In this case an EBC output provides motion reference information directly at pin 2.

The jumper strap from pin 4-5 connects the internally generated position reference command pulses to the input of the motion reference bus driver IC (U29C), wing a PMC to provide its motion reference command information to the Electronic Lineshaft. Moving this jumper to connect pin 3-4 allows the EBC to provide motion reference information directly to the Motion Reference Bus.

3.11 MOTOR INSTALLATION

3.11.1 Motor Use and Environment

The servomotor is designed for use as described below:

- Either horizontal or vertical mounting orientation
- Indoors, clean and dry
- Free from corrosive and/or explosive gases or liquids
- If the location is subject to excessive water or oil, protect the motor with a cover. The motor can withstand a small amount of splashed water or oil.
- Accessible for inspection and cleaning
- ORMEC MTE-Series Servomotor environmental conditions:
  Ambient Temperature: 0° to +40°C
  Storage Temperature: -20° to +80°C
  Humidity: 20% to 80% Relative Humidity (non-condensing)

3.11.2 Coupling the Servomotor to the Load

Good alignment of motor and the driven machine is essential to prevent vibration, reduced bearing and coupling life, or shaft and bearing failures. With direct drive a flexible coupling should be used. Timing belts and chains are also commonly used in servo applications. Bearing loading, particularly axial loading, should be kept to a minimum. Consult the axial and radial loading specifications for the individual servomotor that you have selected.

In either case, it is better to attach the coupling or pulley to the shaft with a clamping arrangement rather than transmit torque through the keyway, because of the shock reversing torque which the servomotor generates. Mechanical devices for this include tapered hubs, split hubs, ringfeder devices, etc.

ORMEC's MTE Series servomotors are designed for face mounting, and the structural integrity of the mounting can be critical to obtaining the maximum performance from your servo application.

OPERATION

4.1 POWERUP

Each time power is applied to the MCS system, or it is reset with the RESET input, it goes through the following operational sequence.

1) The PMC performs diagnostics. Its LED will flash during these diagnostics, and its operation is detailed in the Maintenance and Troubleshooting Section.

2) Initialize a number of parameters such as:
   - **J**og Speed, **A**ccelerations, **I**ndex Distance, and Index **V**elocity;
   - the X, Y, and Z Status Registers
   - the Trace option and Program Memory Write enable (OFF).

   The PMC parameters, including their ranges, units and defaults are detailed in the MPL Software Manual.

3) Check the STOP inputs at the System Axis Interface and the Machine I/O. **If this line is asserted, the rest of the sequence below is aborted,** and the PMC powers up as follows:[4]
   - with no Axis-ID;
   - in Autobaud Mode;
   - without running the @@ Powerup routine, and
   - with the READY Discrete Machine Output line asserted.

4) Test the Axis-ID Configuration Header, and possibly non-volatile memory, and determine its Axis-ID, if any.

5) Check non-volatile memory and determine whether a serial communications baud rate has been previously selected (using the **SB** command).

6) Look in MPL Program Memory for an automatic user defined "powerup routine" (labelled @@) and execute it if found. If the powerup routine is not found, or if it ends with an **E** or **Q** command, the PMC will assert the READY output and wait for a command, either through the Serial Communications Interface or through the Discrete Machine I/O.

4.2 SERVODRIVE

The servodrive is configured for current command input, i.e. the output current of the servodrive, and therefore the output torque of the motor, is proportional to the ±10 volt analog input signal (ICMD). The rated peak current of the servodrive is obtained when ICMD is at approximately ±10 volts.

---

[4] The primary reason to do this is troubleshooting, when the PMC won't seem to communicate via the Serial Communications Interface. See the Maintenance and Troubleshooting Section.

4.2.1 Peak Current Limit Adjustment

The servodrive provides a peak current limit function, which limits the peak current to the current listed in the Specifications, protecting both the servodrive and servomotor. It is adjusted using the peak current limit potentiometer (R28). This 20-turn potentiometer is accessed through the top of the cover of the MCS. This potentiometer is set at ORMEC prior to shipment to provide the peak current indicated in the Specifications (generally fully CW). Some applications may require reducing the peak torque adjustment, which can be accomplished by adjusting this potentiometer in the CCW direction.

4.2.2 RMS Current Limit Adjustment

The servodrive also provides an RMS current limit function, which protects both the servodrive and servomotor and is adjusted using the RMS current limit potentiometer (R36). This single-turn potentiometer is located below connector J1 on the servodrive, and the servodrive must be removed from the MCS-Chassis to access it. This potentiometer is set at ORMEC prior to shipment, and adjusts the servodrive to provide the RMS current indicated in the Specifications. Some applications may require reducing the RMS torque adjustment, which is accomplished with this potentiometer, but should be done only by qualified service personnel.

There is a distinct difference in the peak and RMS current limits! The peak current limit causes the servodrive output current to be hard limited to the limit value, never allowing it to go above that hard limit. Since the RMS current limit is usually lower than the peak current (by about a factor of 2), continued current requirements at a level between the RMS and the peak limits can cause the RMS current to rise above the RMS current limit. When that happens, it causes a "latched" DRIVE FAULT condition of the servodrive, which causes servodrive operation to be disabled. See Section 4.3.

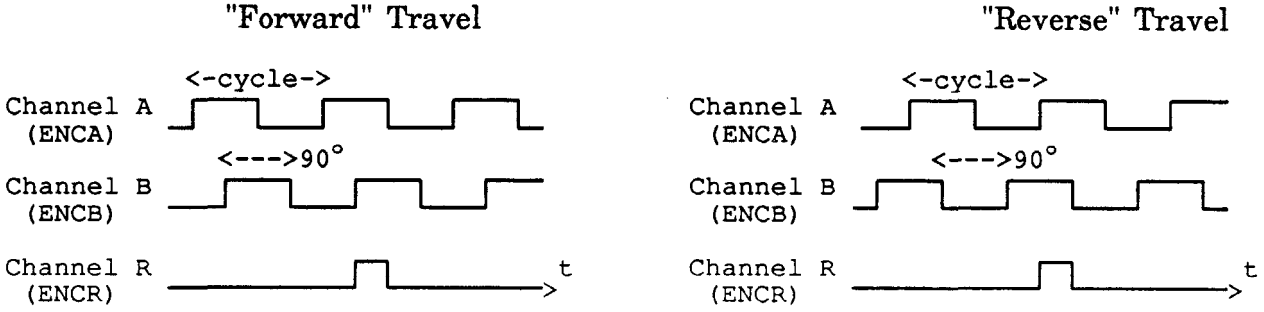
4.2.3 Signal and Balance Potentiometers

The SIG and BAL potentiometers on the servodrive are factory set and require no field adjustment.

4.3 POSITION ENCODER SIGNALS

Optical position encoder signals for "Forward" and "Reverse" travel are illustrated below. Channel A and Channel B are phase quadrature output signals, which allow the Programmable Motion Controller to determine both travel and direction. The "Encoder Resolution" linecount specification of the selected Motor-Tach-Encoder (or separate encoder) specifies the number of "cycles" of the encoder per revolution of the motor, as illustrated below. The PMC in the MCS-920 Series decodes each transition of both encoder channels, yielding a resolution of four times the linecount specification per revolution. e.g. A position encoder with linecount of 1000 yields a positioning resolution of 4,000 cts/rev.

Optical Position Encoder Signals



4.4 REGENERATIVE LOAD CONDITIONS

When conditions exist such that the direction of power flow is from the machine into the motor, the motor acts as a generator. This can occur for a variety of reasons including the following:

1) decelerating the machine faster than it would coast due to friction, this is especially critical at high speeds and with large inertias;
2) using the motor to lower a load that is not counterbalanced, and;
3) using the motor to control an unwind stand for rolls of material, where the tension in the web causes the motor to have to hold back while moving forward.

4.4.1 Overvoltage Protection

The servodrive in the MCS-920 Series uses PWM technology to deliver power to the motor. The switching of the PWM amplifier, in conjunction with the inductances and capacitances in the motor and the output circuitry can cause the power supply to gain voltage as energy is returned to it from the machine. The Bus Power Supply has no mechanism for returning energy to the power line in cases where the motor acts as a generator, and if the bus power supply voltage becomes excessive, the overvoltage protective circuitry will shut off the PWM output transistors, disabling the servodrive, and lighting the DRIVE FAULT indicator.

4.4.2 Applications Strategies

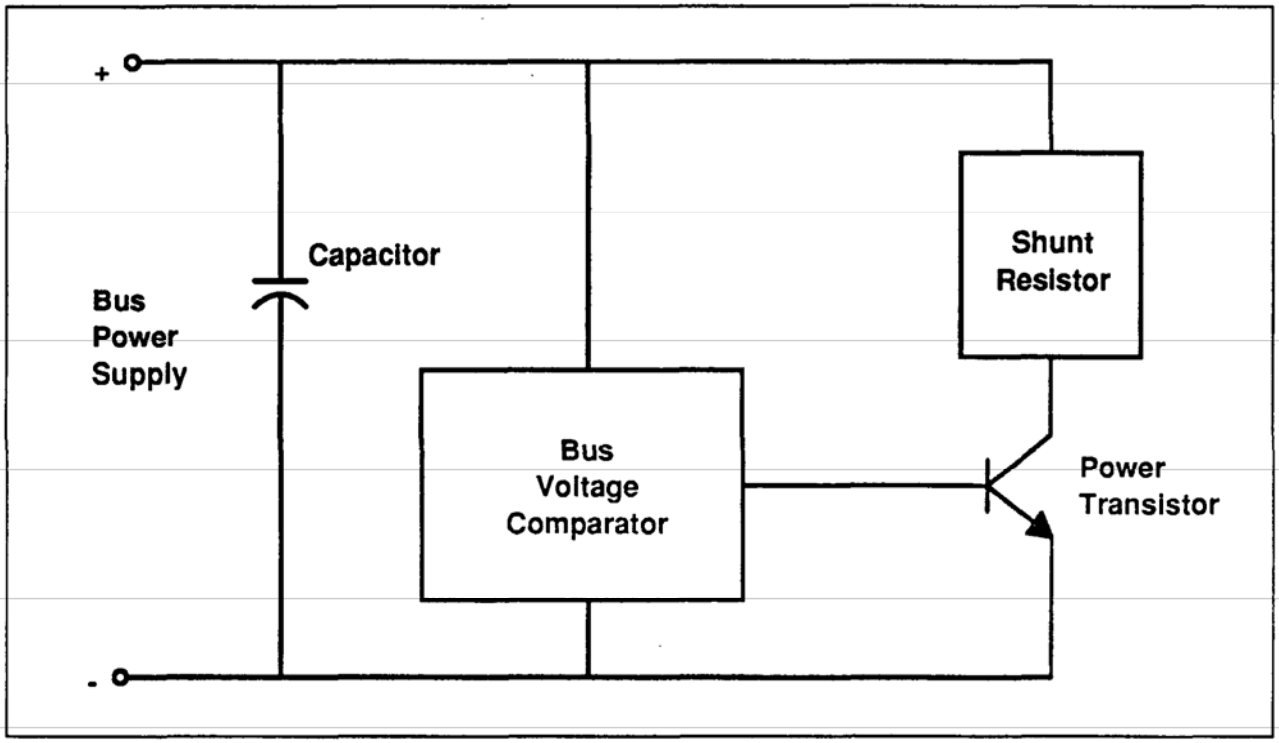
If regeneration is excessive causing the overvoltage fault to occur, the following actions may resolve the problem:

- Slow down the deceleration curve.
- Reduce the current limit.
- Decrease the maximum speed.
- Change the gearbox (design, reverse efficiency or ratio).
- Add an optional external shunt regulator.

4.4.3 Optional Shunt Regulator

A shunt regulator consists of a voltage comparator, a switching transistor and a shunt resistor. When the voltage comparator detects excess power supply voltage (at approximately 185 VDC) it turns on the transistor, dissipating excess energy from the bus power supply capacitor in the shunt resistor. The amount of energy that it can dissipate is dependent on the current capability of the switching transistor and the resistance and wattage of the shunt resistor.

Shunt Regulator Circuitry



4.5 DYNAMIC BRAKING CIRCUITRY

The MCS-920 Series do not include any dynamic braking circuitry, however, if required, it can be added externally by the user using an electromechanical relay with "break before make" contacts and a dynamic braking resistor suitably sized for the desired dynamic braking effects.

4.6 DISCRETE MACHINE I/O OPERATION

The Machine I/O Interface consists of 16 discrete I/O points to interface an MCS system to the machine, a machine console with pushbuttons and indicators, or Programmable Controllers. The MCS includes an Opto-22 I/O rack for an optically isolated, robust and flexible interface. This interface is accessed by MPL programs as well as a number of user selectable standard features, as configured by the PMC's Z-Register.

Operation of the Discrete Machine I/O points is defined by the operation of MPL software in the PMC. The PMC uses "negative" logic for compatibility with Opto-22 style optically isolated I/O modules. The operation of this interface is as follows:

1) To assert an input, voltage appropriate to that particular Opto-22 style module must be applied by the user. In that case:
   - input current will be drawn by the module;
   - the LED will be ON; and
   - the TTL level signal at the Machine I/O connector of the PMC (JM1) will be "low" (between 0 and 0.8 VDC).
2) For the input to be OFF (not asserted):
   - no input current can be present in the Opto-22 style module;
   - the LED will be OFF; and
   - the TTL level signal at JM1 of the PMC will be "high" (between 3.5 and 5 VDC).
3) When the PMC asserts an output;
   - the TTL level signal at JM1 will be "low" (between 0 and 0.8 VDC);
   - the LED will be ON; and
   - the Opto-22 style module will "turn on", conducting current through the load.
4) When the PMC turns an output OFF;
   - the TTL level signal at JM1 will be "high" (between 3.5 and 5 VDC);
   - the LED will be OFF; and
   - the Opto-22 style module will "turn off", not conducting current through the load.

4.6.1 Overtravel Limit Switch Inputs

For applications such as ballscrew or rack and pinion driven X-Y Tables, hardware overtravel limit switch inputs may be desirable.

Inputs IN4 and IN8 can be set up to operate as "negative" and "positive" hardware overtravel limit switch inputs by setting Bit 1 of the X Register. The polarity of these inputs is selectable, to operate with limit switches which are either "closed" (applying voltage to the Opto-module) or "open" when the motion is out of limit. This choice is made using the **SLH** command; the MPL command **SLH0** specifies that the overtravel limit is asserted when the switch is closed, and **SLH1** specifies that the overtravel limit is asserted when the switch is open.

4.6.2 Fault and In-Motion Outputs

Output OUT4 can be programmed to automatically "turn on" whenever a FAULT occurs by setting Bit 2 of the Z Register. This FAULT output indication is cleared at the start of the next MPL command. Output OUT8 can be programmed to automatically "turn on" whenever the PMC is IN-MOTION (i.e. commanding motion) by setting Bit 3 of the Z Register.

4.6.3 Initiating MPL Routines from Hardware

A powerful, yet easy to use, interface to Programmable Controllers (PLCs) can be implemented through the Discrete Machine I/O. The most common approach to this is for the PLC to initiate MPL program **@0** or **@1** by asserting or clearing the SEL input and then, as long as the READY output is asserted, asserting the EXECUTE input. When MPL execution starts, the READY output will be cleared by the motion controller, and if program execution ends, it will be asserted. **The PLC can always abort both the current program and any motion in progress by asserting the STOP input.** The two programs **@0** and **@1** are most commonly used to implement "Auto/Manual" or "Run/Setup" modes of operation.

Alternatively, the IN10-IN80 and SEL digital inputs can be configured to allow the PMC to access up to 31 MPL routines from the Machine I/O interface. The motion routine labels accessible from this hardware addressing scheme are listed in the following table of Hardware Accessible Motion Routines.

Connect 1, 2, 3, 4 or 5 Motion Routine Address lines as shown below depending on the number of motion routines you want to access from hardware for your particular application. To run one of the specified routines, place the appropriate signals on these address lines as indicated in the table, and when the PMC is "READY", as indicated by the READY output, assert the EXECUTE input. General purpose inputs IN10-IN80 continue to be accessible from MPL.

MOTION ROUTINES ACCESSIBLE FROM DISCRETE I/O

Program Label	SEL	IN80	IN40	IN20	IN10	MIS-200 Code (hex)
0	--	d	d	d	d	0d
1	ON	d	d	d	d	1-
	1 address line *					
	SEL	IN80	IN40	IN20	IN10	
@	--	--	--	--	--	00
A	--	--	--	--	ON	01
				1 address line		
B	--	--	--	ON	--	02
C	--	--	--	ON	ON	03
				2 address lines		
D	--	--	ON	--	--	04
E	--	--	ON	--	ON	05
F	--	--	ON	ON	--	06
G	--	--	ON	ON	ON	07
			3 address lines			
H	--	ON	--	--	--	08
I	--	ON	--	--	ON	09
J	--	ON	--	ON	--	0A
K	--	ON	--	ON	ON	0B
L	--	ON	ON	--	--	0C
M	--	ON	ON	--	ON	0D
N	--	ON	ON	ON	--	0E
O	--	ON	ON	ON	ON	0F
		**	4 address lines		**	
P	ON	--	--	--	--	10
Q	ON	--	--	--	ON	11
R	ON	--	--	ON	--	12
S	ON	--	--	ON	ON	13
T	ON	--	ON	--	--	14
U	ON	--	ON	--	ON	15
V	ON	--	ON	ON	--	16
W	ON	--	ON	ON	ON	17
X	ON	ON	--	--	--	18
Y	ON	ON	--	--	ON	19
Z	ON	ON	--	ON	--	1A
[	ON	ON	--	ON	ON	1B
\	ON	ON	ON	--	--	1C
]	ON	ON	ON	--	ON	1D
^	ON	ON	ON	ON	--	1E
	*****	5 address lines			*****	

"ON" => LED ON "--" => LED OFF

"d" => don't care input state

*** Only one address line (SEL) is used with Z Register Bit 4 (10) set**

4.7 PMC SERIAL COMMUNICATIONS

The serial communications interface is provided for motion application development and troubleshooting. In addition, it can be used to coordinate the operation of several PMCs by a host computer, or a PLC using an ASCII BASIC module.

4.7.1 Establishing Communications with the PMC

In its factory default condition, the PMC will automatically set the baud rate of the serial port when a series of "carriage returns" is received[5]. You can send them from your serial communications device by repeatedly pressing the **Return, Enter** or **EXE** key.

Once the PMC determines the baud rate, it will respond with the MPL logon message followed by a "prompt". The logon message will include the version code for the resident firmware, e.g. MPL3.1f + MPL-MATH 1.3a. The prompt will be an "Axis-ID"[6] followed by a } or > character[7]. ORMEC produces a number of "Extended Command Sets", which are sets of commands "added into" our MPL programming language for special applications. If one of these is present in the PMC, the EPROMs will be marked and the logon message will append the ECS number and software revision code.

e.g. MPL3.1f + MPL-MATH 1.3a + ECS999 1.0a

Examples of PMC prompts are:

=} indicates there is no Axis-ID assigned, and the motor is OFF

A> indicates that the Axis-ID is A and the motor is ON

---

[5] For the PMC to perform the autobaud function, it must receive a series of up to eight "carriage return" <cr> characters (ASCII $0D_H$) at the serial communications interface. Each time it receives a character, the PMC checks to see if it is a <cr>. If so the baud rate is set and the autobaud function is complete. If the character is not recognized as a <cr>, the PMC reduces its baud rate by a factor of two, and waits for another character to check. The series must contain enough "carriage returns" to allow the PMC to receive a valid <cr> character at the baud rate of the user serial communications device. Also, the "carriage returns" must not be sent at a rate faster than one every 35 msec, and the complete series must be sent within 2.1 seconds. e.g. If the user terminal communications rate is 9600 baud, three <cr> characters are required; the baud rate starts at 38.4k baud, and after the first character is received, the baud rate is reduced to 19.2k; after the second, the baud rate is reduced to 9600 baud; the third one is recognized since the baud rate is correct, and the PMC's baud rate is then set at 9600 baud.

[6] The Axis ID may be an "equals sign" (=) or a letter from A to Z.

[7] The PMC actually sends a } character (ASCII $7D_H$) if the motor is OFF, and a } character with its "top bit" set (ASCII FD_H) if the motor is ON. ORMEC's MAX-II communications software changes the FD_H to a > before sending it to the screen. If seven data bits are selected in your terminal, it will ignore the top bit and the prompt character will always print as a }. However, if eight data bits are selected in your terminal and the servodrive is enabled, the "}" character of the prompt will change to the character printed by the terminal when an ASCII FD_H character is received.

4.7.2 Setting PMC Serial Baud Rate

PMC Baud rate may be set using the **SB** command, as detailed in the MPL Software Brochure. When this is done, the "autobaud" sequence of carriage returns will no longer be required in order to communicate with the PMC each time it is reset. Instead, the user has the responsibility to communicate at the proper baud rate.

4.7.3 Multi-Axis Serial Bus Communications with PMCs

MCS Systems can be utilized in machines requiring multiple servomotors (multi-axis systems). For multi-axis requirements, ORMEC PMCs provide multi-drop serial bus communications using an RS-422/485 serial bus. The communications protocol on this bus is such that the master machine control computer controls the bus, with all the PMCs operating as talker-listeners. The master computer selects the PMC to communicate with using a two character selection sequence which will be recognized by all the PMCs on the bus, even though they may be running an MPL program and controlling a servomotor.

To use PMCs with multi-axis serial bus communications, they must each:

- **have an assigned address, and**
- **they must have their serial communications interface configured for RS-422/485.**

See the PMC Configuration Jumper section under Installation for instructions with respect to setting Axis-ID in hardware and configuring the Multi-Axis Serial Bus.

Selecting an Axis on the Multi-Axis Communications Bus

Once the serial baud rate has been established, to select a PMC as talker on the bus, the master computer must send a two-character **select axis** sequence. This sequence consists of an:

- Attention Character: **Ctrl]** (ASCII $1D_H$), and the
- Axis-ID: which is an upper case letter from A-Z.

After this sequence is sent, the PMC with the transmitted Axis-ID will be selected as talker, and will respond to serial communications in the same way it would respond if it was not on a Serial Bus.

Assigning an Axis-ID from the Serial Communications Interface

For systems with more than 14 PMCs on a single Serial Bus, the Axis-IDs must be assigned by software and will be stored in non-volatile memory. To do this, connect to a **single** PMC, and execute an **SZ20** command to "write enable" the Axis-ID parameter and the non-volatile baud rate parameter. Then use the **=** (equals sign) command and the **SB** (set baud) command to assign Axis-ID, and communications baud rate if desired. Power down the PMC, and connect in a Multi-Axis Serial Bus before applying power. When it powers up, it will not communicate until the master computer has sent its particular **select axis** sequence.

4.7.4 System Status Polling

The PMC has a feature called System Status Polling, which give the user access to important status information from the serial port during MPL program execution. In addition, the user can both read and write any of the one hundred non-volatile registers in the PMC. A simple yet powerful approach to flexible automation control with MCS systems is to write the applications program in MPL, with important machine operations parameters defined in MPL Registers. The user can then easily change these parameters from an Operator Interface Keypad, or from the serial communications interface to modify the operation of the system.

4.8 MODES OF OPERATION

There are four operating modes of the PMC, which are generally accessed by the **SM** (set mode) command in MPL:

Command	Mode	Description
SM0	IDLE	In IDLE mode, the PMC disables the "servodrive enable" output (SDRVEN) as well as the position and velocity loop compensation circuitry.
SM1	VELOCITY	In VELOCITY mode, the PMC operates as an analog velocity control system only, driving the velocity loop with the analog Feedforward Velocity Reference signal derived by the microprocessor and the motion command circuitry. This voltage is applied to the velocity loop through the feedforward gain adjust (FFGAIN). This mode is useful for setup and troubleshooting of the servo system, as well as motion control applications which change mode from position control to some other feedback source such as force, tension etc.
SM2	POSITION	In POSITION mode, the PMC operates as a digital positioning system, controlling speed and position as a phase lock loop position controller. Digital position reference pulses derived by the Reference Generation Circuitry are summed in the Position Summing Junction (PSJ) and an analog feedforward voltage is applied to the velocity loop. The positioning error is monitored, and if it exceeds the 12 bit capacity of the PSJ ($\pm$2048 counts), the system is returned to IDLE mode.
SM4	MASTER	In MASTER mode, the PMC does not control a servomotor, but instead provides motion information to one or more other PMCs which are controlling servomotors.

4.9 MOTION COMMAND CONFIGURATION OPTIONS

The PMC has a number of configuration options which can be selected by MPL commands using the X, Y and Z Status Registers. ORMEC's MAX-II software on an IBM-PC or compatible allows these options to be selected from a menu.

- **Velocity Range:** The maximum frequency that can be synthesized by the motion command circuitry can be adjusted from 48 kHz to 384 kHz. This sets the maximum speed that the digital position encoder can be commanded to go.
- **Acceleration Profile:** The acceleration profile can be selected to be linear, s-curve, parabolic, or a custom acceleration shape specified by the customer.
- **Motion Bus Master:** Selecting this option causes a PMC to output its digital motion position command pulses to the Motion Reference Bus.

- **Motion Bus Slave:** Selecting this option causes a PMC to synthesize its position command information from pulses provided by the Motion Reference Bus (actually the EXTREF input to the Servo Reference Generation Circuitry shown in Appendix J-1) instead of from the internal crystal clock.

- **Speed Bias:** This option may be selected when the PMC is configured as a Motion Bus Slave, and causes the PMC to transfer 50% of the EXTREF input pulses directly to the Position Summing Junction. The other half of the pulses are used by the PMC to synthesize motion commands, which can either "add to" or "subtract from" the servomotor's commanded speed.

- **External Start Select:** This option causes the PMC's motion commands to be delayed in actually commanding motion until either the Encoder Reference (ENCR) or the Machine Sensor (SENSIN) is received.

- **External Decel:** Selecting this option causes the PMC's deceleration to be initiated when the Machine Sensor (SENSIN) is received.

- **Index Extend:** Selecting this option causes the PMC's deceleration to pause at the "Jog" speed and motion to stop when either the Encoder Reference (ENCR) or the Machine Sensor (SENSIN) is received.

- **External Stop Select:** This option selects either the Encoder Reference (ENCR) or the Machine Sensor (SENSIN) to be used to stop motion during a "**H**ome" command or an "Index Extend".

- **Sharp Jog Stop:** Selecting this option causes the PMC to stop immediately at the end of a jog command or "Index Extend".

4.10 ELECTRONIC LINESHAFTING AND THE MOTION REFERENCE BUS

In a machine with a mechanical lineshaft, various operational stations are attached to the common lineshaft. These stations might be attached with various mechanisms such as gears and timing belts, and the drives for the stations themselves might have simple ratioed motion or they might be more complex, such as crank-drive mechanisms, cams, geneva mechanisms, etc.

Electronic Lineshafting is a term for the operation of multiple digital servomotors in coordination with a common motion reference. Like their mechanical counterparts, the various servomotors might have relatively simple ratioed motion, or they might have more complex motion similar to the more complex mechanical mechanisms.

There are a number of advantages of electronic lineshafting over mechanical lineshafting, including accuracy, but most of them are related in some way to flexibility. ORMEC MCS Systems are designed to operate in electronic lineshafting applications and have a number of features related to this environment. Many of the PMC Configuration options in the preceeding section relate to electronic lineshafting applications.

4.10.1 PMC as Motion Bus Slave

When a PMC is used in its factory default configuration, all motion commands are based on the output of the microprocessor's crystal clock. The PMC's microprocessor commands motor speed by synthesizing a serial pulse train from various ratios of the clock frequency and applying them to a "digital position loop".

When it is configured as a Motion Bus Slave, the PMC commands motor speed by synthesizing a serial pulse train from various ratios of the Motion Reference Bus. The Motion Reference Bus is an RS-485 compatible serial signal which can drive up to 32 PMCs. It can be driven by any PMC (one at a time) by configuring that PMC as a Motion Bus Master. In addition, it can be driven by an external encoder mounted to a non-servomotor driven machine and interfaced by either a QTG-910 or an EBC-900.

When a PMC is configured as a Motion Bus Slave, the units of its motion parameters change, with Velocity specified as a ratio of the Motion Bus. Applicable velocity commands are:

V used to specify speed for relative Index or absolute Go commands in hundredths of a percent, from 0.01 to 100.00%

J used to specify an operating speed, from 0.01 to 100.00%

RG used to specify an operating speed as a function of the exact ratio of two 16-bit integers, e.g. 11/23

RJ used to specify an operating speed, with a resolution of more than one part in 200 million

Acceleration units change also, with acceleration specified in 100s of counts of travel of the Motion Reference Bus.

If the most accurate dynamic positioning is required, the Motion Bus Slave should also receive an analog signal proportional to the frequency on the Motion Reference Bus, to be used by the Velocity Feedforward motion command circuitry. To do this, remove Jumpers 1-2 and 3-4 on Header J7 and place a Jumper from pins 2-4. This is described in the PMC Configuration Jumper Section of the Installation Chapter. If an analog signal proportional to Motion Reference Bus frequency is not used, you should turn off feedforward (command **TF0**), since it's reference is fixed at +2.5 volts.

The Motion Reference Bus receiver and driver lines are tied together on a PMC and so a PMC must be either a Motion Bus Slave or a Motion Bus Master, but not both. **An exception for this exists when a PMC is slaved to an encoder interfaced with an EBC-900** daughter board. This is because the EBC-900 mounts directly on the PMC, and its command signal to the PMC is strapped directly to the EXTREF signal, which must be disconnected from the Motion Bus Receiver. See Appendix J-5 for the schematic of the Motion Bus receiver, driver, and Configuration Header JM20.

4.10.1 PMC as Motion Bus Master

When a PMC is configured as Motion Bus Master, its driver IC (U29) is enabled for output and its digital command pulses (POSREF) drive the Motion Bus. The Master PMC's Velocity Range (see the previous section) must be one range lower than the other PMCs operating as Motion Bus Slaves. e.g. The highest frequency

range the Master can operate in is 256 kHz, assuming the Slaves operate in the 384 kHz range.

If the most accurate dynamic positioning is required of the slaves, then wire the XVELREF signal (TB1-21) to the EXVCMD input (TB1-20) of the slaves. Be sure to set it up to swing positive, with a maximum of 2.5 volts at top speed.

4.10.2 QTG-910 as Motion Bus Master

When a QTG-910 is used as Motion Bus Master, the PMC must be configured with a suitable Velocity Range for the maximum frequency to expect, and the strapping on the QTG. With the QTG configuration jumpers in their *factory default* positions, the maximum operating velocity is 350 kHz, and the PMC should be in the 384 kHz velocity range.

The QTG also provides an analog signal proportional to the encoder frequency and suitable at its terminal block TB1-21, for driving the Slave PMC at TB1-20. Its voltage, with the *factory default* jumper configuration will be +2.5 volts at the maximum frequency.

4.10.3 EBC-900 as Motion Bus Master

When an EBC-900 is used to drive a Slave PMC, the PMC should be operated in a velocity range of 256 kHz or less. In this configuration, other PMC's can be slaved to either:

- the frequency directly from the EBC, or
- the command frequency of the PMC with the EBC daughter board mounted on it, by configuring this PMC as **both** a Motion Bus Slave and a Motion Bus Master.

See the PMC as a Motion Bus Slave Section.

GETTING STARTED

5.1 TEST RUN PREPARATION

For the test run, you need to attach the following items to the motion controller:
1) incoming power, and
2) servomotor & encoder.

DO NOT connect the motor shaft to the driven machine!

Unless your MCS system is pre-programmed, you will also need a serial communications device for startup, programming and test. It is highly recommended that you use one of the following:
1) ORMEC LDS-150 Laptop Development System
2) an IBM-PC or compatible running ORMEC's "MAX-II" motion control development software.

If none of the above are available, configure your terminal device consistent with the serial communications specifications in the Specifications Section.

Before the test run, do the following checks of the servomotor and motion controller and their installation. Correct any problems before proceeding.

5.1.1 Motion Controller Checks

1) Incoming power supplied to the Motion Controller should have its voltage at 115 VAC nominally, **with a MINIMUM of 103 VAC and an ABSOLUTE MAXIMUM of 127 VAC. Check power BEFORE turning on the MCS!**
2) All wiring leads must be firmly connected to terminals.
3) Motor wiring and grounding correct, if other than ORMEC standard MTE Series servomotors.
4) Serial communications device attached to connector JM1 of the PMC.

5.1.2 Servomotor Checks

1) Motor mounting proper.
2) Mounting bolts and nuts tight.
3) Motor rotates freely by hand.
4) Keyway removed, or taped down, for the test.
5) Motor and Encoder Cables properly attached.

5.1.3 Applying Power

1) After checking items above, apply power (switch on top of the MCS).
2) The dual color LED on the PMC circuit board should display the following:
   - RED for a fraction of a second
   - GREEN for 1 second
   - RED for 1 second
   - YELLOW when the on-board diagnostics are successfully completed.
3) The Servodrive's red DRIVE FAULT indicator should be OFF.

If the sequence above is not correct, consult the Maintenance and Troubleshooting Section.

5.2 ESTABLISHING COMMUNICATIONS WITH THE PMC

In its factory default condition, the PMC will automatically set the baud rate of the serial port if a series of "carriage returns" is received. You can send them from your serial communications device by repeatedly pressing the **Return**, **Enter** or **EXE** key. References to this key in this section will be denoted by **[cr]**. See the Serial Communication Operation Section for details on the autobaud feature.

Once the PMC determines the baud rate, it will respond with the MPL logon message and a "prompt", which is an "Axis-ID" followed by a **}** or **>** character.

Examples of PMC prompts are:

=} indicates there is no Axis-ID assigned, and the motor is OFF
A> indicates that the Axis-ID is A and the motor is ON

The prompt indicates that the PMC is operating, and is ready to accept commands.

5.3 TEST RUN

Follow the procedure below to demonstrate basic operation of your system:

1) Type **SM2** to enable the motor in the **Position Mode.** You will hear the Servo Enable Relay "pull in" and observe the motor hold position under power.
2) Type **J5+** to move the motor slowly. It is now "jogging" (running continuously) at a rate of 500 counts per second of your position encoder.

Note that you can stop motion at any time by pressing the [space] bar.

3) If you pressed the [space] bar, type **J5+** again.
4) If you are using MAX-II, press function key **F10**. This displays an "Axis Status" screen and you will see that the current system velocity is 500 counts/second and the system position is growing. Other information is also shown, such as following error and the state of system inputs and outputs.
   - Press the **Esc** key to exit this screen and return to Talk mode.
5) Press the **[space]** bar to stop the motion.
6) Type **I-** to "index" a fixed distance in the negative direction.
7) Type **I?** to find out how far you went (500 counts).
8) Type **V100[cr]** to change the top velocity to 10,000 cts/sec.
9) Type **A500[cr]** to raise the acceleration rate to 500 cts/sec/msec (20ms accel).
10) Type **I10000+** to index 10,000 counts in the positive direction.
11) Type **N0+** to "normalize" the current position to zero.
12) Type **bt[cr]**[8] to demonstrate the motor move back and forth twice.
13) Type **lt9999[cr]** to make it move back and forth continuously. Remember, you can stop the motion by pressing the **[space]** bar.
14) If you're using MAX-II, press **F10** again to view status.
15) Type **h-** to see the motor move to the "once per rev" encoder reference pulse.
16) Type **SM0** to disable the motor by putting it in **Idle Mode**.

At this point, you have demonstrated the fundamental operation of the system.

---

[8] Note that PMC commands are not "case sensitive". i.e. An **Index** command is equally well executed with an **I** or an **i**. Program labels, however, are case sensitive, and the program label **T** is unique from the program label **t**.

5.4 SAMPLE MPL PROGRAM

There is a sample Motion Programming Language (MPL) program in your PMC's non-volatile memory to assist you in getting started. A listing for it is shown on the next page. If you have MAX-II, a "downloadable" copy is also included on the distribution disk in the \WORK directory with the filename MCS-920.MPL.

Note that the disk copy of the program is identical to the listing, and contains all the comments and blank lines as shown. When downloaded by MAX-II, all blank lines will be removed, as well as all text which follows [space] characters or [tab] characters on a line. This feature allows a well documented file to exist on your development computer without using up all the limited memory (8k bytes) on the PMC. Note that in the program listing which follows, the comments are preceeded by one or more **[space]** or **[tab]** characters, followed by one or more * characters.

The sample program consists of an automatic Powerup routine, labelled @@, as well as a Tuning program, labelled **@T** and a number of subroutines. The Powerup routine sets up the system Gains and Status Registers as they were adjusted at the factory. You actually ran one of the subroutines, labeled **@t** during the test run when you typed **bt** and **lt9999**. This routine is located on the last page of the listing.

The Tuning Program is shown to illustrate good MPL programming style and help you fine tune your MCS system after the motor is attached to the load. See the sections after the Program Listing on "Editing the Program" and "Tuning the System" for an explanation of the program.

```
**********************     Sample MPL+MATH Program     **********************
**********************          MCS-920.MPL            **********************

**********************       PROGRAM LABELS USED       **********************
** @   Power-up
** A   Adjust tuning
** F   Feedforward tuning
** G   Get following error samples routine
** L   Load Status Registers & Gains from non-volatile Registers
** M   Tuning routine main Menu
** P   Position tuning
** R   Register initialization
** T   Tuning routine initialization
** V   Velocity tuning

** c   clear screen
** d   display error samples
** f   feedforward tuning entry point
** g   get following error samples initialization
** h   homing
** i   Index
** k   keystroke read
** p   position tuning entry point
** r   restore motion parameters
** s   save Status Registers & Gains in non-volatile Registers
** t   toggle tuning motion
** v   velocity tuning entry point

**********************       MPL PROGRAM LISTING       **********************

@@_Powerup  *** @@ marks the start of the automatic powerup routine **********
fR:R90<>100     initialize Registers first time or if different version
R90=100         set software version identifier
vR7             set default index velocity (R7)
aR8             set default acceleration (R8)
iR9             set default index distance (R9)
jR10            set default jog speed (R10)
fL              load Status Registers & Gains from non-volatile Registers
ri@1:cT0        input device set to PMC main port (device 1), character mode
?@1             output device set to PMC main port (device 1)
e               end of Powerup routine

@Registers  *** Register Assignment and Initialization (and documentation) ***
 ** R0          character input; used in Menu selection
 ** R1          'V'elocity or 'P'osition function select
 ** R2          tuning gain amount
R3=126          gain raise factor of 1 db (2 implied decimal places)
 ** R4          tuning compensator amount
 ** R5          'G'ain or 'C'ompensator function select
 ** R6          temporary storage of position compensator value
R7=167          set index velocity register to 16.7 kHz (500 RPM w/2000 cpr)
R8=278          set accel register to 278 Hz/msec (867 rad/sec² w/2000 cpr)
R9=2000         set index distance register to 2000 counts (1 rev w/2000 cpr)
R10=5           set jog speed register to 500 Hz (15 rpm w/2000 cpr)
 ** R11         temporary storage of current feedforward gain
R12=1000        set home offset register to 1000 counts
 ** R38         pointer for following error samples
 ** R39 - R78   following error samples
```

```
@Load       *** Load Status Registers & Gains from non-volatile Registers ****
sxR87           set X-Register to the value in Register R87 (0-255)
syR88           set Y-Register to the value in Register R88 (0-255)
szR89           set Z-Register to the value in Register R89 (0-255)
tpR91           tune Position gain to the value in Register R91 (0-255)
tvR92           tune Velocity gain to the value in Register R92 (0-255)
tfR93           tune Velocity Feedforward to the value R93 (0-255)
txR94           tune eXternal gain to the value in Register R94 (0-255)
tcpR95          tune Position Compensator to the value in R95 (0-F Hex)
tcvR96          tune Velocity Compensator to the value in R96 (0-F Hex)
e

@save       *** save status registers & gains in non-volatile registers ******
R87=sx          save current value of X-Register (R87)
R88=sy          save current value of Y-Register (R88)
R89=sz          save current value of Z-Register (R89)
R91=tp          save current Position gain (R91)
R92=tv          save current Velocity gain (R92)
R93=tf          save current Velocity Feedforward gain (R93)
R94=tx          save current eXternal gain (R94)
R95=tcp         save current Position Compensator (R95)
R96=tcv         save current Velocity Compensator (R96)
e

 ******************        MAIN PROGRAM ENTRY LABEL        ******************
 ******************     Start Program by typing BT[cr]     ******************

@Tune       *** Tuning routine initialization *********************************
R7=v            save current motion parameters
R8=a
R9=i
R10=j
R11=tf

@Menu       *** Tuning routine main Menu ***************************************
fc              clear screen
?"~~~~~~~~~~TUNING~STATUS:",
t?              display current tuning values
?               display blank lines
?
?"Select:~~Velocity~~Position~~Feedforward~~Index~~Home~~Save~~eXit?~",
fk              read command input key
r?R0='V'|R0='P'|R0='F'
                IF tuning command
v50                 set motion parameters for a small move for tuning
a2000
i2000
j400
?                   display blank lines
?
f(R0+020)           execute "lower case" command routine
r:R0='X'        ELSEIF eXit command
fr                  restore original motion parameters
e                   exit this program
r:R0='I'|R0='H'|R0='S'
                ELSEIF other valid command
f(R0+020)           execute "lower case" command routine
r?              ENDIF
bM              branch back to main Menu
```

```
@vel_entry  *** Velocity tuning entry label ***********************************
?"***~~VELOCITY~~GAIN~~ADJUST~~***",
R1='V'          select 'V'elocity function
R5='G'          select 'G'ain function
r?tv<2          IF Velocity gain too low
R2=2                set tuning gain (R2) to lower limit
r:              ELSE
R2=tv               set tuning gain (R2) to current Velocity gain
r?              ENDIF
R4=tcv          set tuning compensation (R4) to current Velocity Compensator
sm1             set velocity mode (mode 1)

@Velocity   *** Velocity tuning routine ***************************************
tvR2            set Velocity gain to tuning gain (R2)
tcvR4           set Velocity Compensator to tuning compensation (R4)
r?R11=0         IF the Feedforward gain at program entry was 0
R11=100             set the Feedforward gain register to 100
r?              ENDIF
tfR11           drive velocity loop with Feedforward gain set at program entry
bA              Adjust tuning and read input and continue

@pos_entry  *** Position tuning entry label ***********************************
?"***~~POSITION~~GAIN~~ADJUST~~***",
R1='P'          select 'P'osition function
R5='G'          select 'G'ain function
r?tp<2          IF Position gain too low
R2=2                set tuning gain to lower limit
r:              ELSE
R2=tp               set tuning gain (R2) to current Position gain
r?              ENDIF
R4=tcp          set tuning compensation (R4) to current Position Compensator
sm2             set position mode (mode 2)

@Position   *** Position tuning routine ***************************************
tpR2            set Position gain to tuning gain (R2)
tcpR4           set Position Compensator to tuning compensation (R4)
tf0             disable Feedforward function
bA              Adjust tuning and read input and continue

@ffwd_entry *** Feedforward adjustment entry label ****************************
R6=tcp          save Position Compensator value (R6)
tcp0            turn off Position Compensator
tf0             turn off Feedforward gain
sm2             set position mode (mode 2)
j+              start positive continuous motion

@Feedforwrd *** Feedforward adjustment loop label *****************************
tf+             increase Feedforward gain
d30             wait for system to settle from gain change
?"Following~Error:~",te!,"~~with~TF",tf
bF:te!>0&tf<255 repeat until following error is negative
j*              stop motion
tf-             decrease Feedforward gain for a positive following error
t?              display current tuning values
d2500           allow time to read the display
tcpR6           restore Position Compensator value (R6)
e
```

```
 ****************************** Sub-functions ********************************

@Adjust      *** Adjust tuning ************************************************
t?                  display current tuning values
ft                  toggle the tuning motion
?"~~~~~+~~~~~Gain~~Compensator~~eXit~?~",
fk                  read command into R0
r?R0='+'            IF increment value command
r?R5='G'                IF 'G'ain selected
R2=(R2*R3+50)/100           raise gain
r:R5='C'                ELSEIF 'C'ompensator selected
R4=R4+1                     raise break frequency
r?                      ENDIF
r:R0='-'            ELSEIF decrement value command
r?R5='G'                IF 'G'ain selected
R2=(R2*1000/R3+5)/10        lower gain
r:R5='C'                ELSEIF 'C'ompensator selected
R4=R4-1                     lower break frequency
r?                      ENDIF
r:R0='X'            ELSEIF e'X'it command
tfR11                   restore Feedforward gain
e                       exit (to main Menu)
r:R0='G'            ELSEIF 'G'ain adjust command
R5=R0                   select 'G'ain function
?                       display blank lines
?
?"****~~",R1:c,"~GAIN~~~ADJUSTMENTS~~****",
r:R0='C'            ELSEIF 'C'ompensator adjust command
R5=R0                   select 'C'ompensator function
?                       display blank lines
?
?"**~",R1:c,"~COMPENSATOR~~ADJUSTMENTS~**",
r?                  ENDIF
b(R1)               branch to 'V'elocity or 'G'ain function

@g_er_entry *** get following error samples entry label **********************
R38=39              point to first error sample register (R39)

@Get_error  *** get following error samples loop label ***********************
R(R38)=te!          R(pointer) = following error
R38=R38+1           point to next error sample register
lG39                get next error sample (total of 40 samples)
R38=39              point to first error sample register (R39)
?                   display blank line

@disp_error *** display error samples loop label *****************************
?R(R38):8d,         print following error sample
R38=R38+1           point to next error sample register
ld39                display next error sample (total of 40 samples)
e

@cls        *** Clear screen routine ***************************************
?"^[[2J",           ANSI escape sequence to "clear the screen"
e
```

```
@home       *** Home Routine ***************************************************
sm2             set position mode (mode 2)
h-              home to the nearest encoder reference in "-" direction
d,              delay until reference found and motion stops
iR12+           index to the offset home position
d250,           delay until motion is settled
n0+             normalize the tuning offset and set absolute zero
e

@index      *** Index routine **************************************************
sm2             set position mode (mode 2)
d50             delay for system to stabilize
n0+             normalize the tuning offset and set absolute zero
fr              restore original motion parameters
i+              command positive index motion
fg              get and display following error samples during the motion
d1000           allow time for person to read the display
e

@key        *** Read keystroke routine *****************************************
riR0:1c;        read single key input command
r?R0>='a'&R0<='z' IF 'a' <= keystroke <= 'z'
R0=R0&95            convert to upper case
r?              ENDIF
?R0:c,          display key input command
e

@restore    *** restore motion parameters **************************************
vR7
aR8
iR9
jR10
e

@toggle     *** toggle tuning motion *******************************************
i+              command small positive move
d300,           delay for motion to stop and settle
i-              command small negative move
d300,           delay for motion to stop and settle
lt1             repeat motion toggle once (two toggle motions)
e
```

5.5 EDITING THE PROGRAM BUFFER

If you are using MAX-II with an IBM-PC or compatible, it is recommended that most of your editing be done with a text editor on it, and the subsequent file downloaded to the PMC. This edit can be accessed directly from MAX-II in the "Application" pull-down menu. However, the PMC also includes a simple text editor, allowing the program buffer to be edited interactively.

We are now going to use the program listed above to gain some experience editing the program buffer. In the sequences listed below, **bold** print indicates the sequence that you type, and the regular print is the information sent by the PMC. See the Motion Programming Language Booklet for a full description of the editing commands.

The PMC has a "write protect" feature which write protects MPL program memory each time the unit is reset or powered down and so before you can change anything, you must "write enable" the PMC.

Type **SW1** to set write-enable ON.[9]

Type **Pt** to enter "Program mode" with the cursor at the beginning of the "t routine"; The PMC responds by displaying the first line as shown.

@t If you have an IBM-PC or compatible with ORMEC's motion development software (MAX-II), you will be able to use the cursor positioning **[arrow]** keys, as well as the delete key for editing.

Review the program buffer a line at a time by typing successive line-feeds, usually LINEFEED or LF on most ASCII terminals. If you can't find a linefeed key, a linefeed can be typed by typing a "control J" (depress the CTRL key and while holding it, press the J key).

Press **[down arrow]** or linefeed **[lf]** to move the cursor down a line

i+ Press **[down arrow]**

d300, Press **[down arrow]**

i- Press **[down arrow]**

d300, Press **[down arrow]**

lt1 Press **[right arrow]** or **[tab]**; The cursor is now on the 1 and so you can overtype it. Overtype the 1 with a 2. You can also back up a character by typing a **[left arrow]** or **[backspace]**. If you back up beyond the first character in a line the cursor will move back to the end of the previous line. The **[up arrow]** key will move you up one line. In this way, you can move the cursor around the program buffer until you find the area you want to overtype, and then modify it.

Press the ESCAPE **[Esc]** key, to end program editing and return to the interactive mode, signified by the =} "READY" prompt.

---

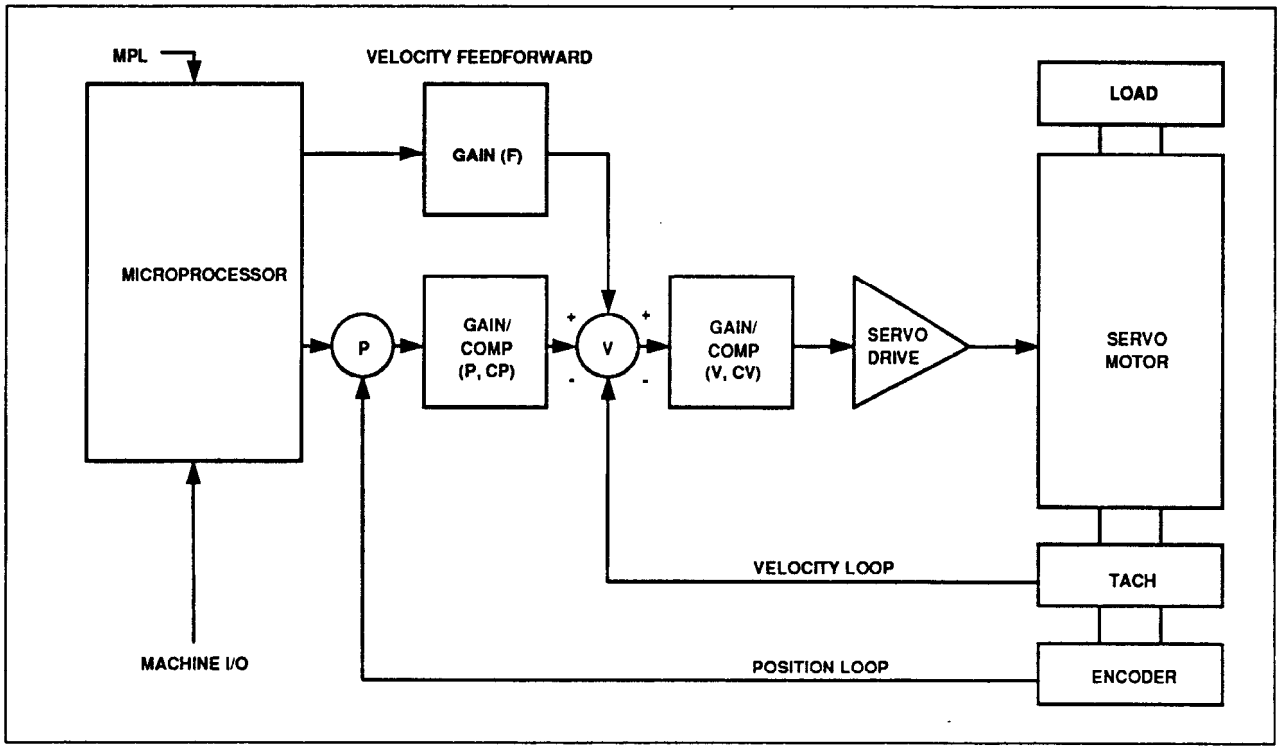
[9] Never put an SW1 command in the Program Buffer, or it will override the intent of the "write protect" feature, which is to protect the Program Buffer from being inadvertently changed.

5.6 TUNING THE SYSTEM

Your system has been factory tuned for loads up to the inertia of the servomotor. However, MCS systems can control servomotors with loads attached which are much greater than the inertia of the motor. To optimally match your motor and MCS system to your application, the software selectable tuning parameters must be adjusted after the load is attached to the servomotor.

The Tuning Program has been provided to assist you in matching the system to your particular application. Once you have made your changes, the tuning program will also help you save them in non-volatile registers for future use.

The illustration below shows the PMC gain and compensation adjustments available to the user to tune the system.



The recommended strategy for tuning the PMC is to perform the following adjustments in the order listed below:

VELOCITY LOOP:	Adjust the Velocity Loop Gain **V** Adjust the Integral + Proportional Compensator **CV**
POSITION LOOP:	Adjust the Position Loop Gain **P** Adjust the Integral + Proportional Compensator **CP**
FEEDFORWARD:	Adjust the Velocity Loop Feedforward Gain **F**

Type **BT[cr]** to run the tuning program.

5.6.1 Adjusting the Velocity Loop Gain

This adjustment (accessed in MPL by **TV**) is provided to adjust the gain in the analog velocity loop. It is adjustable from a wide range of 1 to 255 (48 db) to be compatible with a wide range of inertias, and varying performance criteria.

At the Main Menu of the Tuning Program, do the following:
- Press **V** to adjust the velocity gain.

The **@v** Velocity Loop tuning routine will put the system in **Velocity Mode** with an **SM1** command, show the current gains and toggle the tuning motion[10]. After this, it will prompt for a keystroke.
- Press **V** again to repeat the motion without changing the loop gain.
- Press **-** to lower the gain by 1 db and repeat the motion.
- Press **+** to raise the gain by 1 db and repeat the motion.

The recommended approach is to raise the velocity loop gain until the system either "overshoots" (becomes underdamped) or it breaks into resonance (buzzes).

For many systems, it is sufficient to raise the gain until the system buzzes, and then lower it until it stops. For critical operations, attach an oscilloscope to the tachometer test point. This test point **TCH** on the PMC is buffered and filtered, and is the middle of three test points located in the corner of the PMC near TM1. See Appendix H.

As you raise the velocity loop gain, observe the tachometer signal and, when it overshoots or the system exhibits resonance by oscillating or "buzzing", reduce the gain until it stops. Note that the PMC reports its gain and compensation parameters during the tuning process, due to the program line **T?** in the **@V** routine.

The velocity loop "rise time" (time to go from 10% to 90% of full value) for most servo systems during this test should be between 1 and 15 msec, as long as the system is operating in "small signal" mode. To be operating in small signal mode, the commanded velocity must be small enough so that the motor voltage and current do not saturate (limit). Normally a speed of between one and five revs per second will not saturate the servo system.

5.6.2 Adjusting the Velocity Loop Integral + Proportional Compensator

Once the Velocity Gain is adjusted for no overshoot, but with a reasonably fast rise time (under 10 msec, and probably under 5 msec), the Velocity Loop *Compensator* **CV** break frequency should be adjusted. The recommended approach again is to raise it until the system either "overshoots" (becomes underdamped) or it breaks into resonance (buzzes).

---

[10] For initial tuning of the servo, the **@s** subroutine sets the top velocity to 5,000 counts/second, the acceleration rate to 2,000 cts/sec per msec (2.5 msec accel time), and the move distance to 2000 counts. If the servomotor has a 500 linecount encoder, the PMC decodes 2,000 counts per revolution and the motor velocity is set for 2.5 revolutions per second (150 RPM). Since the **@V** Velocity tuning routine uses the factory preset feedforward gain, the speed in velocity mode will be approximately correct. The times for the signal will be exact however, since they are based on the microprocessor's crystal oscillator.

Continuing from where you left off adjusting the Velocity Gain:
- Press **C** to begin making adjustments to the Velocity Compensator
- Press **-** to lower the compensator break frequency and repeat the motion.
- Press **+** to raise the compensator break frequency and repeat the motion.

5.6.3 Adjusting the Position Loop Gain

To adjust the Position Loop Gain, go back Main Tuning Menu by pressing **X**.

At the Main Menu of the Tuning Program, do the following:
- Press **P** to adjust the position gain.

The **@p** Position Loop tuning routine will put the system in **Position Mode** with an **SM2** command, show the current gains, turn off the Feedforward gain with a **TF1** command, and toggle the tuning motion. After this, it will prompt for a keystroke.
- Press **P** again to repeat the motion without changing the loop gain.
- Press **-** to lower the gain by 1 db and repeat the motion.
- Press **+** to raise the gain by 1 db and repeat the motion.

The recommended approach is to raise the position loop gain until the system either "overshoots" (becomes underdamped) or it breaks into resonance (buzzes).

With your oscilloscope still attached to the tachometer signal, raise or lower the gain. Watch the tach signal for overshoot or system resonance. Again, for many systems, it is sufficient to raise the gain until the system buzzes, and then lower it until it stops.

When the position loop gain is adjusted, return to the Main Tuning Menu by pressing **X**.

5.6.4 Adjusting the Position Loop Integral + Proportional Compensator

Once the Position Gain is adjusted for no overshoot, but with a reasonably fast rise time (under 30 msec, and probably under 15 msec), the Position Loop *Compensator* **CP** break frequency can be adjusted. This compensator is optional, and usually only used in systems which must "track" a reference position accurately while at speed. It can negatively affect the settling time in high speed indexing operation.

If you decide to use it, the recommended approach again is to raise it until the system either "overshoots" (becomes underdamped) or it breaks into resonance (buzzes).

Continuing from where you left off adjusting the Position Gain:
- Press **C** to begin making adjustments to the Position Compensator
- Press **-** to lower the compensator break frequency and repeat the motion.
- Press **+** to raise the compensator break frequency and repeat the motion.

When complete, press **X** to go back to the Main Tuning Menu.

5.6.5 Adjusting the Velocity Feedforward Gain

- Press **F** at the Main Tuning Menu.

The **@f** Feedforward Gain tuning routine stores the position compensator value in Register R6 temporarily, and then turns off the compensator with a **TCP0** command. It then turns the Feedforward off with a **TF0** command, puts the system in **Position Mode** with an **SM2** command, and runs the motor "forward" with a **JF+** command.

It then enters a loop **@F** where it continues to raise the feedforward gain until the position error goes negative. Each iteration of the loop prints the Following Error and Feedforward Gain to the screen.

You will observe during this process that the position following error will be reduced each time FFGAIN is raised. When the following error goes negative, the system is actually leading the commanded position, because the feedforward gain is too high. At that point the program exits the loop, turns the Feedforward gain down by 1 with the **TF-** command, displays the final gains, restores the position compensator value, and returns to the Main Menu. This approach may result in feedforward gains somewhat higher than actually desired, and if it does, simply turn it down interactively with the **TF-** command.

5.6.6 Using the Index Command

The Tuning Program Main Menu includes an Index command which will initiate an index and then call the **@g** routine, which will gather 40 following error samples and report them to the screen. The samples are collected at a rate of approximately every 10 msec, and so the program is useful for viewing the position error during an index. You can exit the Tuning Program, set up an Index with the **A, V,** and **I** commands, and then run the Tuning Program again with a **BT** command to see the Index performance.

5.6.7 Using the Home Command

The Tuning Program Main Menu includes a Home command, which runs the **@h** Home routine. This program finds the nearest encoder reference pulse in the negative direction, and then indexes to an offset home position stored in Register R12. Try running it, then exiting the Tuning Program and examining Register R12 with a **?R12[cr]** command. Then try modifying Register R12 by typing **R12=2000** (or some other number of your choice). When you've changed it, run the Tuning Program again with a **BT** command to see the new Home position.

5.6.8 Saving your Tuning Adjustments

The Tuning Program Main Menu includes a Save command. You may have noticed that when exiting and restarting the Tuning Program the gains were always unchanged from the last time. That's because the Tuning Program always worked with the PMC tuning parameters. These parameters are not in non-volatile memory, however, and are "loaded" from non-volatile registers R91-R96 by the **@L** routine on powerup. See the @@ automatic powerup routine, which calls **@L** to "load" these parameters.

Pressing the **S** key at the Tuning Program Main Menu runs the **@S** routine, which "saves" the tuning parameters in the non-volatile Registers. It also "saves" the **X, Y** and **Z** Status Registers.

If you have MAX-II, it includes a number of standard setup files, which are accessed with the PMC Configuration menu. These files have filenames such as MTE-350A.SET. For the MTE series motors, the first seven characters of the filename indicate the series of ORMEC motors. e.g. The MTE-350 series includes the MTE-350, MTE-351 and MTE-352. The eighth character indicates the encoder resolution according to the following code:

- A indicates a 500 linecount encoder (2,000 cts/rev).
- B indicates a 1000 linecount encoder (4,000 cts/rev).
- C indicates a 1250 linecount encoder (5,000 cts/rev).
- D indicates a 2500 linecount encoder (10,000 cts/rev).

Loading a configuration file will restore the factory standard gains and settings for the X, Y and Z Status Registers. You may also save configuration files of your own on disk using MAX-II.

5.7 CREATING A POWERUP ROUTINE

When you prepare your application program, you will want to determine its operation on powerup by creating your own @@ automatic powerup routine. It is highly recommended that you do this by expanding on the @@_Powerup routine shown, along with the **@L** and **@R** sub-routines, and ending at the **@T** Tuning Program. For MAX-II users, you should start with the diskcopy of MCS-920.MPL.

Note that the powerup routine calls the **@R** register initialization routine to initialize non-volatile registers if Register R90 is not equal to the "User Software Version Code." Note also that the **@R** routine sets R90 equal to 100, signifying "version code 1.00".

Following this standard in your powerup routine will allow different versions of software, when loaded into a PMC, to automatically detect on powerup (or reset) whether or not they had been run on this PMC previously so that they can initialize their non-volatile registers if appropriate.

Holding the STOP line active at either the System Axis Interface or the Machine I/O Interface during powerup will cause the powerup routine to be ignored by the PMC. See the Powerup discussion in the Operation Section.

THEORY OF OPERATION

6.1 INTRODUCTION

An ORMEC MCS-Series Motion Control System includes a Programmable Motion Controller (PMC), a Servodrive and Discrete Industrial I/O, as well as all necessary power supplies. This system is a user programmable universal motion controller which can be used to control high-performance servomotors in a wide variety of applications. It is a "language based" system whose actual operation in any given application is completely defined by a "user program" written in ORMEC's high-level Motion Programming Language, MPL+MATH.

6.2 PMC ARCHITECTURE AND INTERFACES

6.2.1 PMC Architectural Overview

The PMC System Architecture is described in Appendix J-1. You may refer to that diagram for the rest of this Overview.

At the heart of the PMC is the microprocessor (CPU), which runs an interpretive high level Motion Programming Language called MPL+MATH, or just MPL. The machine language code that defines MPL, as well as the fundamental operation of the PMC is stored in EPROM in the "Standard Firmware" socket and the "Optional Firmware" socket. The operation of the language in the unit is extendable by ORMEC to meet specific user applications needs, and if an extension is ordered, it is included in the EPROM in the Optional Firmware socket.

The actual operation of the PMC in any application is defined by a user changeable program written in MPL. MPL is an interpreter which translates high-level user commands into the desired action. This is much like the language in a programmable calculator, or the BASIC programming language on a larger computer. User commands can be executed interactively or from a Program Buffer (Motion Control Program Memory), which consists of 8k bytes of non-volatile RAM.

With a calculator, the ultimate desired action is the solution of a math formula as input by the user. With a PMC, the ultimate desired action is programmable motion for flexible automation control, and the language provides a rich array of commands, functions and features to achieve that end. They are described in the section on the language below.

For the user's MPL program to create programmable motion, the PMC requires a number of other elements in addition to the CPU and memory. Those additional elements are listed below:

- **Encoder Interface Circuitry:** This circuitry interfaces the feedback position encoder, to let the servo sub-system know how far the motor has moved.
- **Servo Loop Subsystem:** This circuitry, in conjunction with the servodrive and motor, closes the digital position loop, and the analog velocity loop creating a high-performance position regulator.

- **Motion Command Circuitry:** This circuitry operates under the CPU's direction to generate the necessary motion command information for the Servo Loop Subsystem.
- **Serial Communications Interface:** This interface is used to develop application programs and also to interactively command and/or monitor motion while the system is performing the application.
- **Machine I/O Interface:** This interface is used to interface with the driven machine and/or user electronics such as a Programmable Controller.

6.2.2 Incremental Position Encoder

ORMEC MCS Series motion control systems are configured to use an incremental position encoder to measure changes in load displacement and provide position feedback. Most MCS based digital positioning systems use optical incremental encoders such as those installed on ORMEC standard servomotors, however any type of position encoder that produces TTL compatible phase quadrature outputs is acceptable. It is common to use an MCS system with a linear encoder.

An encoder with phase quadrature outputs has two digital square wave outputs, which are displaced in phase by 90 degrees, and switch between 0 VDC and +5 VDC. These two outputs are commonly called "Encoder Channel A" (ENCA) and "Encoder Channel B" (ENCB), and the direction of motion determines the phase relationship between them. i.e. If for "forward" motion, Encoder Channel A leads Encoder Channel B by 90 degrees, then for "reverse" motion, Encoder Channel A will lag Encoder Channel B by 90 degrees. See Appendix I for a diagram showing the relationship between ENCA and ENCB for forward and reverse motion.

For more information on incremental position encoders, consult a textbook or literature from a position encoder manufacturer. An excellent source of information on the subject is the book Techniques for Digitizing Rotary and Linear Motion by Dynamics Research Corporation of Wilmington, MA.

The resolution of the position encoding is determined by the number of lines per unit distance that cause output pulses. **Linecount** is a term commonly used by manufacturers of incremental position encoders, and it refers to the number of line pairs of the physical encoding device per unit distance. e.g. lines per revolution, or lines per inch if the system is using a linear encoder.

6.2.3 Encoder Interface Circuitry

Refer to the diagram of the PMC Motor Loop Interface in Appendix J-4 for the following discussion. The encoder is interfaced to the PMC through differential receiver circuit U1. Input resistance pads of 22k ohms, with 2.2k ohms to common provide high input impedance and overvoltage protection. Resistor network RN3, which is socketed, provides a 2 VDC nominal switching point. The outputs of the receiver circuit go to the quadrature decoder circuitry shown in Appendix J-1.

The quadrature decoder circuit interprets the phase quadrature encoder input channels to determine the direction and distance traveled. The PMC's quadrature decoder circuit utilizes "4x multiplication" circuitry, which means that every transition of the two encoder channels is utilized to create a forward or reverse motion feedback pulse, as appropriate. These signals (ENC FWD and ENC RVS) are illustrated in Appendix I.

Because four counts of digital position information are derived from each encoder cycle, the effective resolution of a digital position control system utilizing a PMC is "4x" the "linecount" of the incremental position encoder. e.g. Using an encoder with a linecount of 1000 lines per revolution results in a digital position control system with a resolution of 4000 counts per revolution.

The quadrature decoder circuitry performs two other important functions, noise rejection and synchronization. For noise rejection, it insures that each encoder channel input meets a minimum time criteria, before it is accepted. The digital filter to do this is clocked by the high frequency timing circuitry which provides the quadrature decoder function. If a pulse long enough to meet the minimum time criteria is received, the decoding circuitry interprets it as a count from the encoder, and decodes whether it is in the forward or reverse direction. The synchronization function is performed so that the encoder feedback information will not "collide" with position command information in the Up-Down counter used for the Position Summing Junction.

The encoder pulses from the quadrature decoder are subtracted from the position reference information in the digital position summing junction, which in turn has a count in it proportional to the position error of the control system.

6.2.4 Servo Loop Subsystem

Refer to the PMC System Architecture drawing in Appendix J-1 and the PMC Analog Architecture in Appendix J-2 for the following discussion. A simplified diagram of the Servo Loop Subsystem is found in the section on Tuning the System in the Getting Started Chapter.

The servo loop subsystem essentially consists of two feedback control loops or "regulators". The first is a digital position loop, and the second is an analog velocity loop. Both of these loops are controlled by hardware, and so there is no "loop update time" associated with the system. Also, both of these loops have software controlled loop gains and compensators which are accomplished with multiplying DAC (Digital to Analog Converter) and switched capacitor filtering technology. Therefore the microprocessor is in control of the loop parameters, but does not have to use any computing power to maintain loop regulation.

The digital position loop is closed in the Position Summing Junction, which is a 12-bit Up-Down counter which compares serial forward or reverse command pulses with the encoder feedback pulses. The resulting number of counts in this counter is the position error in the system. The goal of the position loop subsystem is to minimize this error, regulating the feedback position to be equal to the commanded position. It does this by commanding the velocity loop to move the motor when a position error is detected.

To do this, the position error count is converted to an analog voltage by a Digital to Analog Converter (DAC). The resulting "Analog Position Error" signal (APOSERR) is amplified by UA2B, whose output (POSERR) is available at the POS test point. It is then processed by the position compensation amplifier (UA2C). Here the position loop gain (PLGAIN) is adjusted over a range of 1 to 255 (48 db) and an "integral + proportional compensator" can be enabled and adjusted by setting PLCOMP. Position loop gain is set with the **TP** command, and the position loop compensator break frequency is set with the **TCP** command.

The resulting analog output is then applied to the velocity loop summing junction in amplifier UA5D. Other analog inputs summed there are the analog feedforward velocity command (FFVELREF), the analog velocity feedback (HVTACH or LVTACH) and an auxiliary velocity command signal (VCMD).

In ORMEC's MCS-920 Series systems, an analog DC tachometer is used to sense motor velocity and provide velocity feedback. In the MCS-S Series, this signal is derived from the encoder quadrature signals on the motor position encoder. The velocity feedback signal (HVTACH or LVTACH) is summed with the feedforward velocity reference (FFVELREF) and the compensated position error signal resulting in the current command signal (ICMD). This signal is the primary input to the servodrive, which is configured as a current amplifier, with a $\pm$ full current (torque) output for $\pm$10 volts at the input.

Velocity loop gain (VLGAIN) is adjusted over a range of 1 to 255 (48 db) and an integral + proportional compensator can be enabled and adjusted by setting VLCOMP. Velocity loop gain is set with the **TV** command, and the velocity loop compensator break frequency is set with the **TCV** command.

Velocity feedforward gain (FFGAIN) is adjusted over a range of 1 to 255 (48 db), and is adjusted with the **TF** command. The value of this gain affects the amount of speed command sent directly to the velocity loop by the microprocessor.

6.2.5 Motion Command Circuitry

The CPU in the PMC uses the Servo Reference Generation Circuitry to command motion. This circuitry has two outputs, which interface to the position and velocity summing junctions.

The motion command circuitry output to the position summing junction is a serial pulse train (POSREF) and a DIRECTION signal. In the Position Summing Junction this serial pulse train is counted in an Up-Down counter and compared with actual movement of the position encoder. The frequency of this pulse train is proportional to desired speed and the number of counts sent is equal to the desired distance to move the motor. It must also be synchronized so that it will not "collide" with the encoder feedback information in the Up-Down counter.

The motion command circuitry output to the velocity summing junction is an analog voltage proportional to the desired speed. In the velocity summing junction, this voltage is compared to the voltage of the tachometer. If in position mode, the velocity summing junction also receives an analog input from the position loop which adjusts the velocity to minimize position error. Use of this input allows the motion control system to travel at exactly the right speed (on average) with absolutely no accumulative error with respect to the digitally commanded position.

The motion command circuitry has the following inputs:

- the crystal clock on the CPU,
- a serial pulse train from the Motion Reference Bus (EXTREF),
- the "machine sensor", and
- the "once per revolution" encoder reference.

In its factory default configuration, the PMC's CPU and motion command circuitry work together to digitally synthesize position command information from the crystal clock. This creates very precise output frequencies, resulting in highly

accurate velocity control. Motion command information to the velocity loop is simultaneously synthesized from a +2.5 volt voltage reference.

The Motion Reference Bus provides a means for synchronizing the motion of the servomotors with other servomotors or with non-servomotor controlled equipment. By selecting a configuration option, the PMC can digitally synthesize its position command information from externally provided information at the Motion Reference Bus. To use this feature, the user drives the Motion Reference Bus with a serial frequency proportional to the desired speed. This can be done by user electronics, another PMC, or by an encoder interfaced through an ORMEC QTG-910 or EBC-900.

The PMC's Motion Command Circuitry features "double buffered hardware", which allows the CPU to set up a motion command which will actually begin when either the "machine sensor" or the "once-per-revolution" encoder reference is received. This feature can be used to initiate acceleration or deceleration within one encoder count of an external event, and might commonly be used to cause a servomotor driven mechanism to precisely register its position to a reference mark sensed by an Optical Registration Sensor.

The PMC has a number of motion command configuration options which can be selected by MPL commands using three status registers (X,Y and Z). A complete description of them appears in the Operation Chapter.

6.2.6 Summary of Position System Operation

To initiate a position change of a certain number of counts of the position encoder, the CPU causes the Motion Command Circuitry to send the desired number of pulses to the Position Summing Junction through the Position Reference (POSREF) inputs at a rate proportional to the desired velocity. Simultaneously, it applies an analog velocity feed-forward voltage (FFVELREF) to the velocity summing junction to directly command velocity. Since the velocity loop reacts faster than the position loop, this speeds up the reaction of the system and minimizes the error required in the Position Summing Junction.

When the velocity loop senses a difference in the commanded velocity and the actual velocity as measured by the tachometer, it outputs a current command (ICMD) to the servodrive, which causes the motor to generate torque in the appropriate direction. Movement of the motor results. This movement is sensed by the tachometer and encoder and transmitted back to the PMC. The polarities of the forward/reverse command pulses and the decoded encoder pulse are such that the contents of the digital Position Summing Junction is reduced as the encoder feedback is received (negative feedback). If the input pulse train continues at a constant rate long enough for the transient to decay, the encoder feedback pulse train will be forced to have the same frequency as the command pulses.

When the PMC determines that it's time to decelerate, it ramps the command frequency down to zero, commanding the system to stop. When the load moves a distance equivalent to the number of command pulses, the error value in the Position Summing Junction goes to zero, causing the system to stop. Because of the PMC's "4-times" encoder multiplication circuitry, a system using an encoder with a "linecount" of 2500 lines per revolution will move 1/10,000 of a revolution for each command pulse received. Since the position loop remains active as long

as the system is still in position mode, a holding force related to the position loop gain and bandwidth will be present as required to hold proper output position.

6.3 MOTION PROGRAMMING LANGUAGE

6.3.1 Language Overview

The PMC's Motion Programming Language is architected to provide a logical, consistent and easy to use set of commands which specify high performance motion. The result is a calculator like language which provides the application designer a great deal of freedom and power with which to solve unique motion control application needs.

MPL utilizes a set of one and two character commands followed by an optional argument (number) and one or more single character command terminators. The brevity of the language provides a great deal of "power per keystroke", making it quick and easy to use. In addition, it requires minimal operation overhead when operating from the internal Program Buffer or when communicating via the serial port to a host computer.

Like a programmable calculator, or the BASIC programming language, commands can be executed interactively or combined in a "program" (or "routine") for future automatic execution. These routines are stored and edited in the "Program Buffer" using the Program command.

Because execution time is an important factor in most high performance motion control applications, the language is designed to operate quickly, with many commands requiring less than a millisecond and the longer commands requiring a few milliseconds. To keep this time to a minimum, but provide sufficient range and accuracy, arithmetic operations are accomplished with 32-bit integers.

In addition, the PMC is architected to be able to service the real time requirements of commanding motion in the background, while simultaneously running this interpretive language. This feature allows the PMC to perform necessary calculations to set up future motions while simultaneously performing positioning tasks, enabling the user or host computer to talk to the PMC while motion is being controlled.

6.3.2 Language Function and Features Overview

MPL provides a rich array of commands, functions and features to provide sophisticated, high-performance, motion control in an easy-to-use, integrated environment. These features are outlined below:

- **Parametize desired motion:** The following commands are provided for parametizing motion.
  - **A** sets an acceleration rate
  - **I** sets a relative (index) distance to be moved
  - **V** sets a top velocity to use during an index motion
  - **J** sets a top speed to be used for a continuous (jog) motion
  - **H** sets a top speed to be used when the motor is seeking its "home position"

N normalizes an absolute position counter to a value

- **Initiate a motion:** The following commands are provided for initiating various types of motion.
  I causes the motor to move a relative (index) distance
  G causes the motor to go to a particular absolute position
  J causes the motor to accelerate to and run at a continuous (jog) speed until commanded otherwise
  H causes the motor to move to its nearest home reference

- **Synchronize motion program operation with a motion:** Once an MPL program initiates a motion, that motion is automatically executed by the Programmable Motion Controller and MPL continues to operate. This is a powerful feature allowing MPL to be used for a variety of control purposes. e.g. to measure a number of motion parameters such as speed, position, error or distance to go; to respond to discrete inputs, turn on or off discrete outputs, etc. Many of the commands in the MPL language can be synchronized with the controlled motion by inserting a synchronization character in the command. Commands can be synchronized with the start of a motion, the end of the acceleration, the start of the deceleration, or the end of the motion.

- **Store a motion program in non-volatile memory for automatic execution:**
  P opens the Program Buffer for examination or editing

- **Program flow is controlled by a number of commands and functions:** The commands below can be used to control program flow, and the conditional commands have access to the discrete I/O as well as the use of 20 arithmetic and logical operators in MPL MATH.
  @ establishes a program label for reference as a start point or a subroutine
  B causes program execution to conditionally or unconditionally transfer to a program label
  • an IF-THEN-ELSEIF-ELSE structure supports structured programming
  L causes program execution to loop a desired number of times (four layers of loops are supported)
  F causes a function (subroutine) to be conditionally or unconditionally called (three layers of function calls are supported)

- **Integrated MATH functions:**
  • Twenty arithmetic and logical operators are fully integrated into the PMC by more than 50 motion control related functions.
  • One hundred 32-bit non-volatile registers are supported by MPL MATH.

- **Operator Interface Commands:** Operator interface is supported by print and input commands which deal with decimal, string and hexadecimal I/O.

- **PMC Configuration Commands:** The system has many software configurable elements and features including the loop tuning parameters, the synchronization of motion with other machinery, use of the discrete I/O and host communications mode.

MAINTENANCE & TROUBLESHOOTING

7.1 DISCRETE LED STATUS INDICATIONS

7.1.1 Drive Fault Indicator and Servodrive Reset

The servodrive output transistors will be disabled and the red DRIVE FAULT indicator will be ON under any of the following conditions:

- motor or motor leads shorted from either side of the armature to ground
- output leads of the servodrive are shorted together before the inductor
- servodrive bus power exceeds the maximum (203 VDC nominal)
- servodrive bus power less than the minimum (45 VDC nominal)
- RMS current exceeds the servodrive rating
- excessive ambient or heatsink temperature
- the sum of the low voltage ±12 VDC power supplies is less than 22 VDC

The SERVODRIVE RESET pushbutton will allow the protective circuitry to reset assuming the fault is gone. If the fault is gone, the servodrive will then turn off the DRIVE FAULT indicator and allow the output transistors to be enabled. They can still be disabled externally if either the PMC or external user circuitry is shorting D-ENABLE (on TB5) to ground.

7.1.2 PMC Powerup Diagnostics and LED Operation

Each time the PMC powers up or is reset using the RESET' input, the CPU performs powerup diagnostics. The diagnostics test RAM and EPROM memory, and as they operate they toggle the two color LED on the edge of the PMC.

Diagnostic LED normal operation after powerup:

RED	while RESET' is active
GREEN	while the RAM test operates (about 1 sec)
RED	while the EPROM test operates (about 1 sec)
YELLOW (red/green)	**normal system operation**; This toggling, which looks YELLOW, is done at the 4 msec rate of the on-board real time clock. Since the LED is driven directly by the CPU, its continued toggling is dependent on the CPU's processing of the real time clock's interrupts. Therefore, if the CPU stops operating the LED will be either RED or GREEN.

Diagnostic LED error conditions:

GREEN 1 second & RED 1/3 second	RAM failure. Return PMC to ORMEC
RED 1 second & GREEN 1/3 second	EPROM failure. Return PMC to ORMEC
RED 1 second & GREEN 1 second	Program Buffer Checksum Error or Lithium Battery Low. See below:

Program Buffer Checksum Error:

PMC Powerup Diagnostics include a "Checksum" calculation and comparison test of the contents of the non-volatile RAM. This is a method of checking the validity of the program which you entered in the Program Buffer. It examines the program currently in memory to insure that a RAM failure resulting in a changed program has not occurred.

When a Checksum failure is detected, **MPL program execution will be halted.** In addition to the Diagnostic LED flashing, the "MPL Prompt" will be changed to "-}". **Program operation cannot continue until the Checksum Error is cleared using the NK* command.**

The LED should then return to the standard YELLOW color. The user should then either check or download the program buffer to insure that its contents are proper. To insure that the non-volatile RAM is operating properly, powering the PMC down and back up after this update is advised. If the non-volatile RAM is used to store either an Axis-ID, or a Baud Rate, then these two items should also be verified with the **SB!** or **=!** as appropriate.

The most common cause of this problem is powering down the PMC while in the process of writing an MPL program. The reason for this is that the "Checksum" is updated at the end of each Program command. If the program is changed without ending the Program command properly, then the checksum will not agree with the current contents of the Program Buffer and a false "error" will be detected the next time the PMC is powered up. **The proper technique for terminating the Program Command is to use the escape key [Esc], which will cause the PMC to return to the "prompt" level.**

Lithium Battery Low:

If the LED is flashing RED for one second and GREEN for one second, but the unit still operates normally, then the Lithium Battery is low, and a replacement part should be ordered from ORMEC.

7.2 TROUBLESHOOTING GUIDE

7.2.1 Cannot Communicate with Motion Controller

1) The baud rate may be set in the PMC. If you have MAX-II, simply execute the PMC Find Axis command, which will try all baud rates and find all Axis-IDs at the highest baud rate found. If you don't have MAX-II, either set your terminal for the appropriate baud rate, or assert the STOP input as power is applied to the PMC. This will cause the PMC to execute its "autobaud" function. This will allow the user to communicate with the axis and to query or change the baud rate setting. Refer to the Powerup discussion in the Operation Section or the MPL Software Manual for more details.

2) A PMC may have been given a software Axis-ID. If you have MAX-II, simply execute the PMC Find Axis command, which will try all baud rates and find all Axis-IDs at the highest baud rate found. Axis-IDs are retained in the PMC even after the power has been turned off. If you do not know the

Axis-ID, asserting the STOP input while power is applied to the PMC will not allow the PMC to support multi-serial communications (the PMC will not have an Axis-ID). The PMC will then not have to be selected in order to establish communications.

3) If your PMC is configured for RS-422 communications and you are using an ORMEC Serial Bus Interface (SBI) to convert the RS-422 signals to RS-232 signals, make sure that the appropriate jumpers are present on the PMC Header J5. See the discussion on PMC Configuration Jumpers in the Installation Section. You also need the jumpers that provide the necessary power to drive the circuits on the SBI.

4) The LED near the serial connector on the PMC must be a yellow or orange color. If this LED is red or green, check all of the configuration jumpers on the PMC against the configuration drawing provided with your system. If all of the configuration jumpers are correct, check the +5 VDC and the + and - 12 VDC voltages at **TB2** on the PMC. If you are unable to correct this problem after checking the PMC Installation and Operation Manual, call ORMEC.

5) If your PMC is configured for Multi-Axis communications and has been given an Axis-ID, you must use a capital letter to select the PMC. All Axis-IDs must be specified in capital letters.

7.2.2 Motor will not turn

1) Make sure that the bus power from the transformer to the amplifier chassis is hooked up. Also make sure that the circuit breaker supplying power to the transformer is turned on.

2) Check to make sure that all necessary power is turned on. If a circuit breaker has tripped, check all wiring before applying power. Check the fuses in the servo amplifier chassis. Again, if they are blown, check all wiring, especially motor wiring. If you continue to have problems, call ORMEC.

3) When the PMC is put into velocity mode or position mode (SM1, SM2), the servo-drive enable signals (SDRVEN and SDRVEN*) are asserted. SDRVEN is asserted when it is at approximately 4 VDC and SDRVEN* is asserted when it is at approximately 0 VDC. Check to make sure that the D-ENABLE, F-ENABLE and R-ENABLE signals on TB5 are at +12 VDC.

7.2.3 Motor runs away at high speed with PMC in Velocity-mode (SM1)

1) Check the wiring of the tachometer leads to the PMC, and check the wiring of the motor leads to the current smoothing choke.

2) If all connections are correct, reverse the wiring of the tachometer going into the PMC.

7.2.4 Motor drifts with PMC in Position-Mode (SM2)

1) Check the value of the tuning parameters by typing the command **T?** If the value of the position gain is zero (TP0), then the motor may turn, as in velocity mode. Tune the system as described in the Tuning discussion in the Getting Started Section above.

2) If you have determined that the tuning values are correct, the encoder feedback may be reversed. Exchange the wiring of the encoder signals going to encoder inputs **ENC A** and **ENC A'** on TB1 of the PMC.

7.2.5 Motor response is very sluggish after system powerup

1) Check the value of the tuning parameters by typing the command **T?**

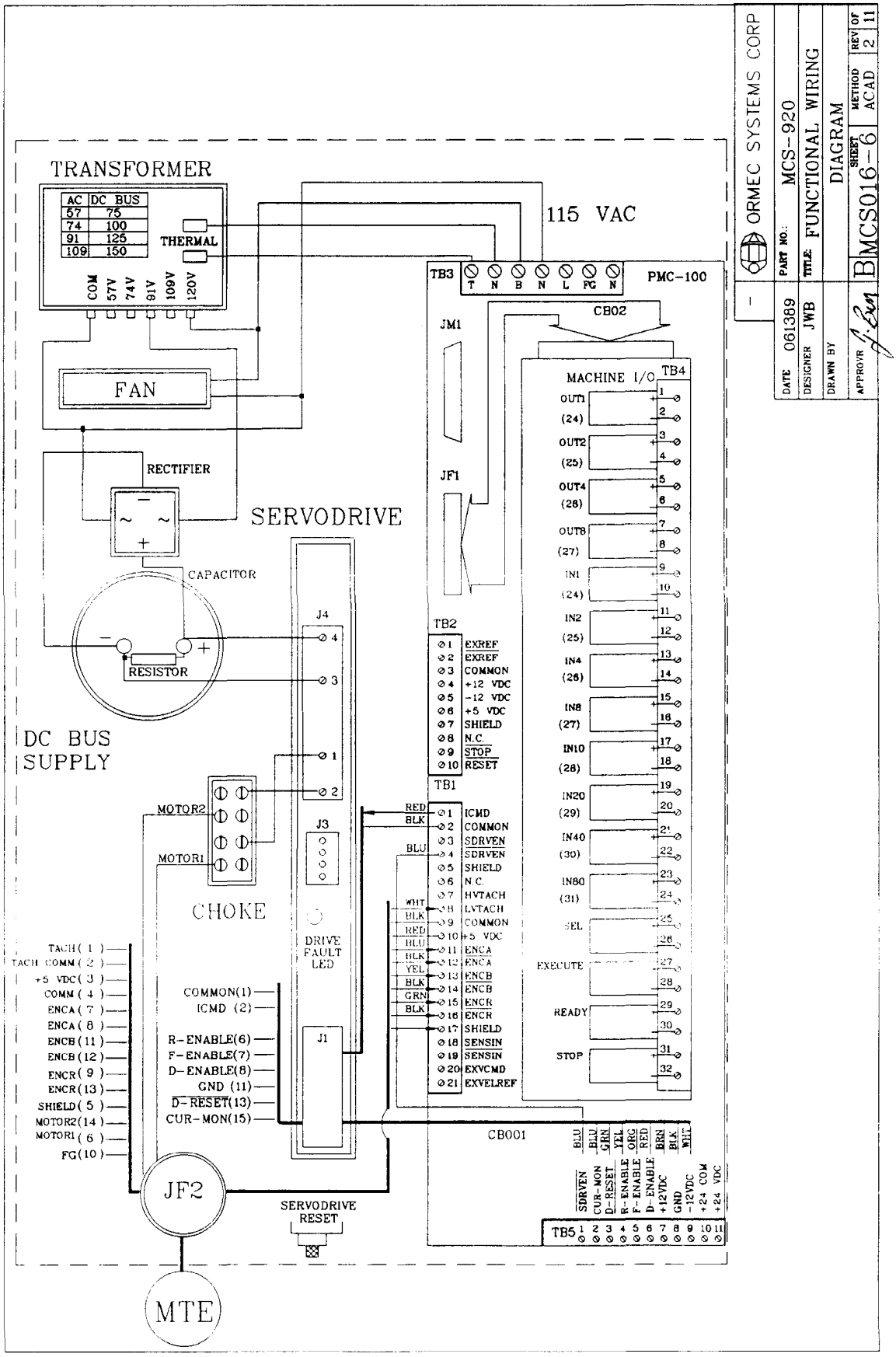
   If the tuning parameters are reported as:

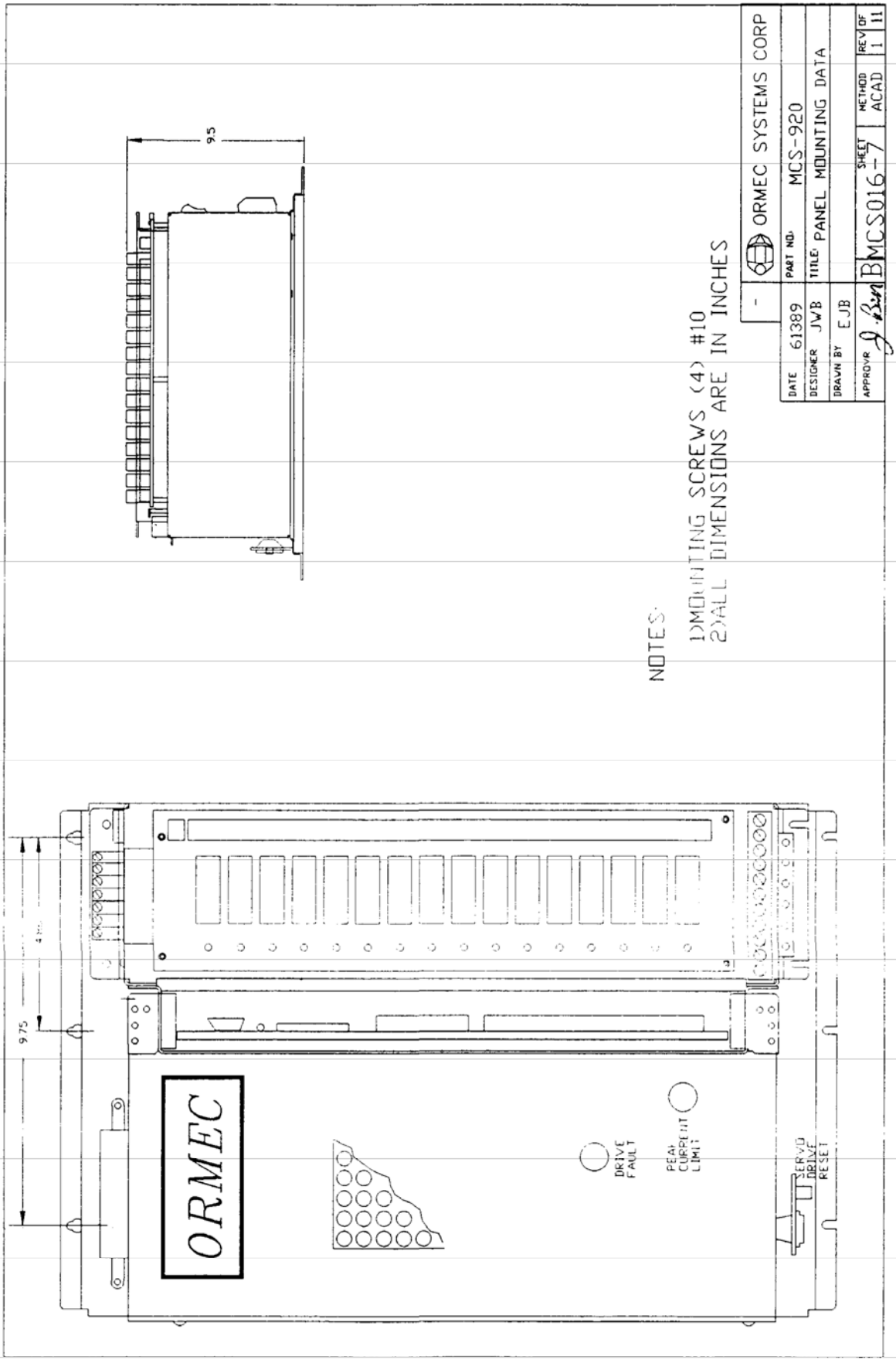
   P=2 V=2 F=0 X=0 CP=00 CV=00

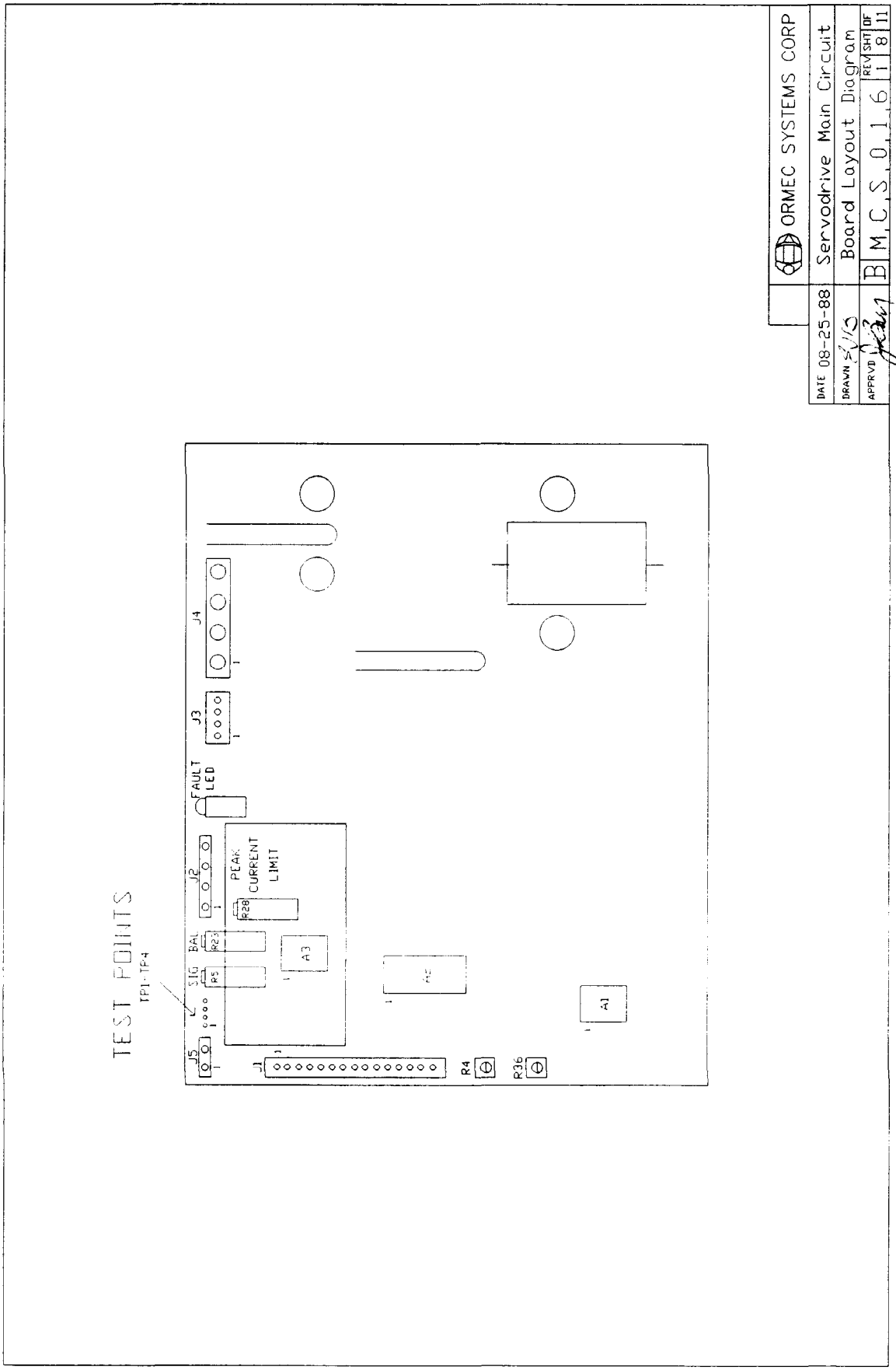
   it is very likely that the powerup routine has not been run.
   - This is probably because the powerup routine (named @@) is not present. If a routine is found named @@, check to make sure that the tuning values are actually set in it.
   - Also, if serial communications are received or the STOP input to the PMC is asserted when the power to the PMC is turned on, the powerup routine will be prevented from running.

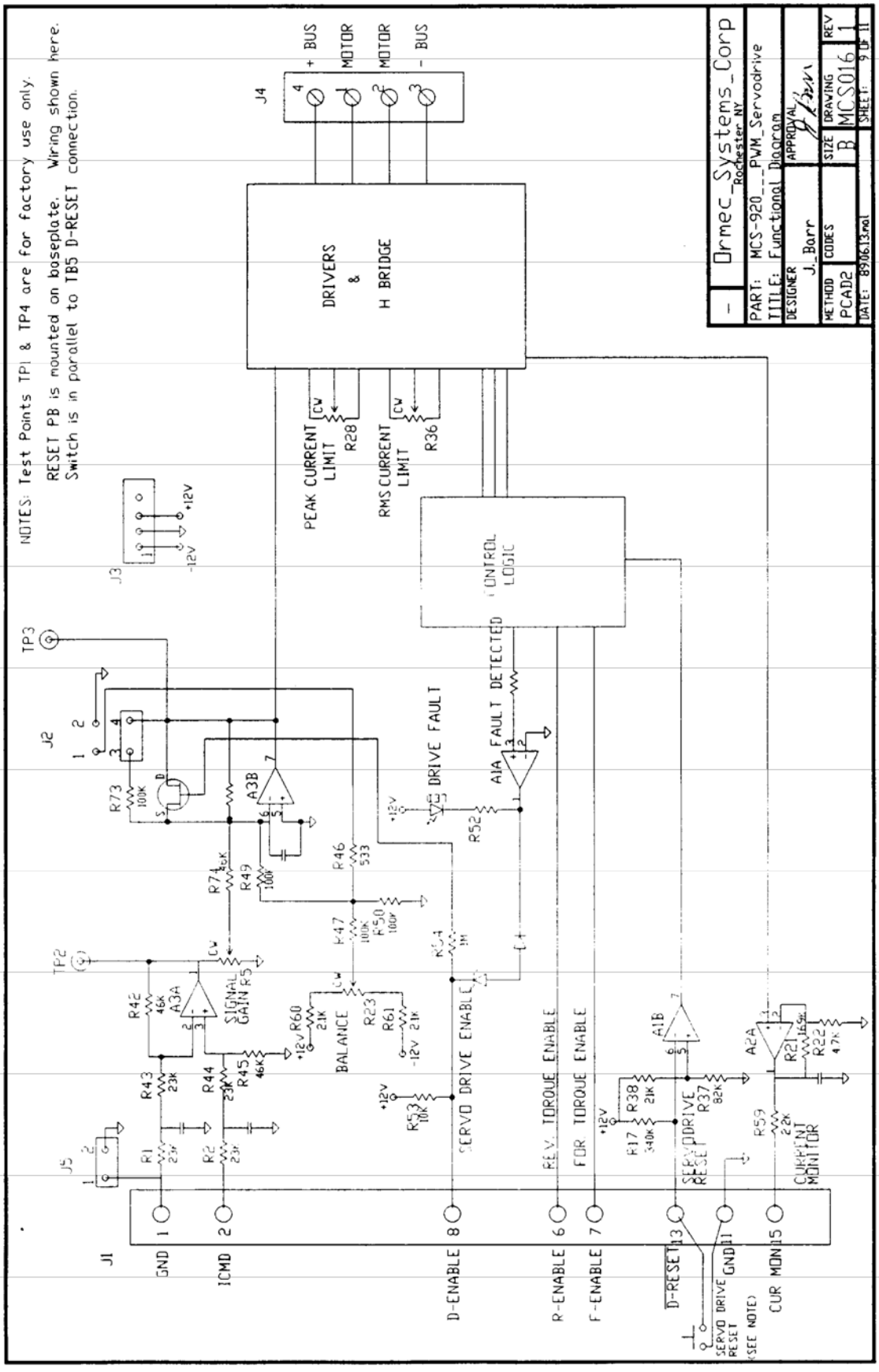
   Another reason for sluggish operation is just that the position (TP) and/or velocity (TV) gains are low. In this case, re-tune the system as directed in the Getting Started Section.

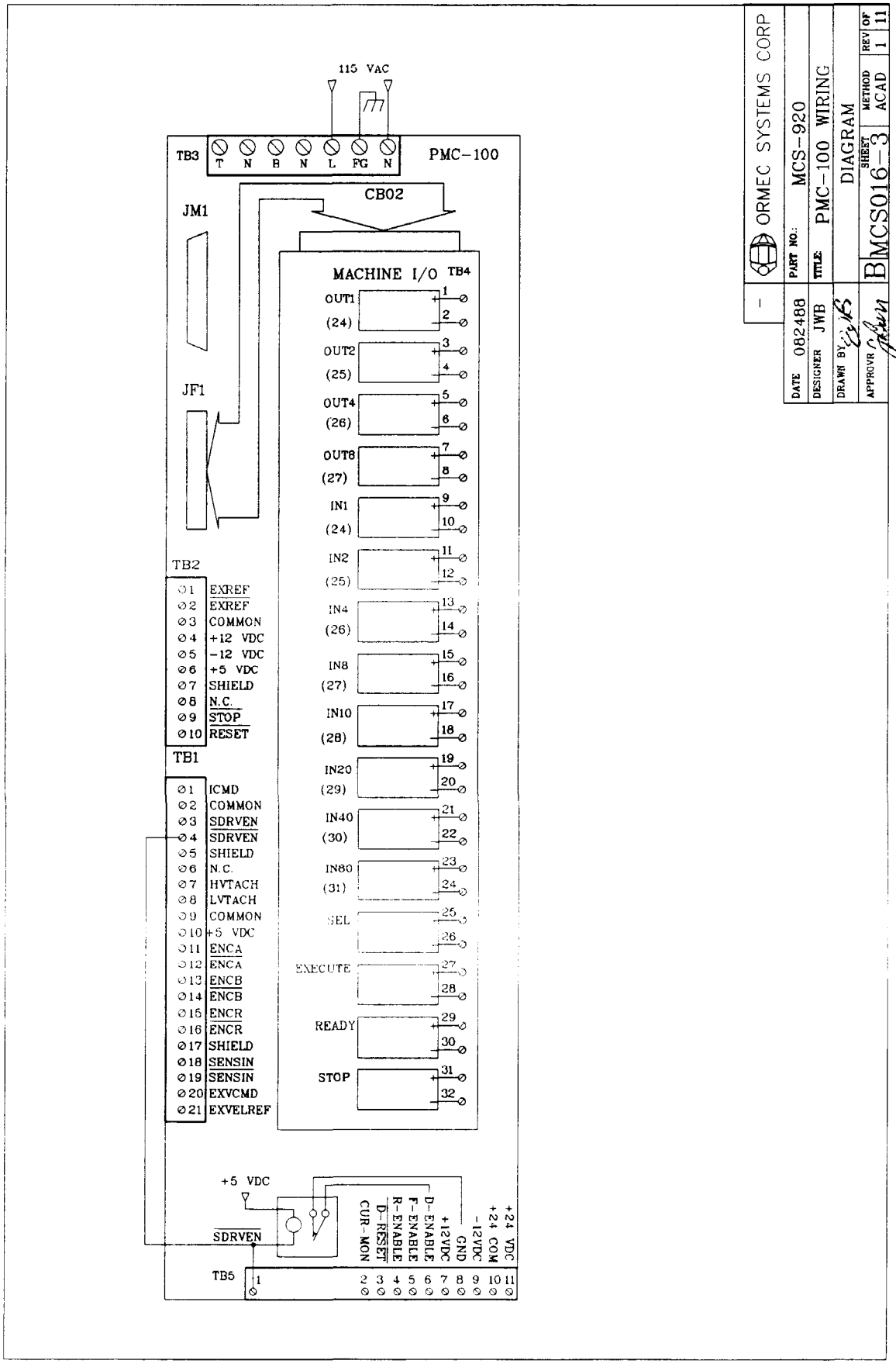
INDEX

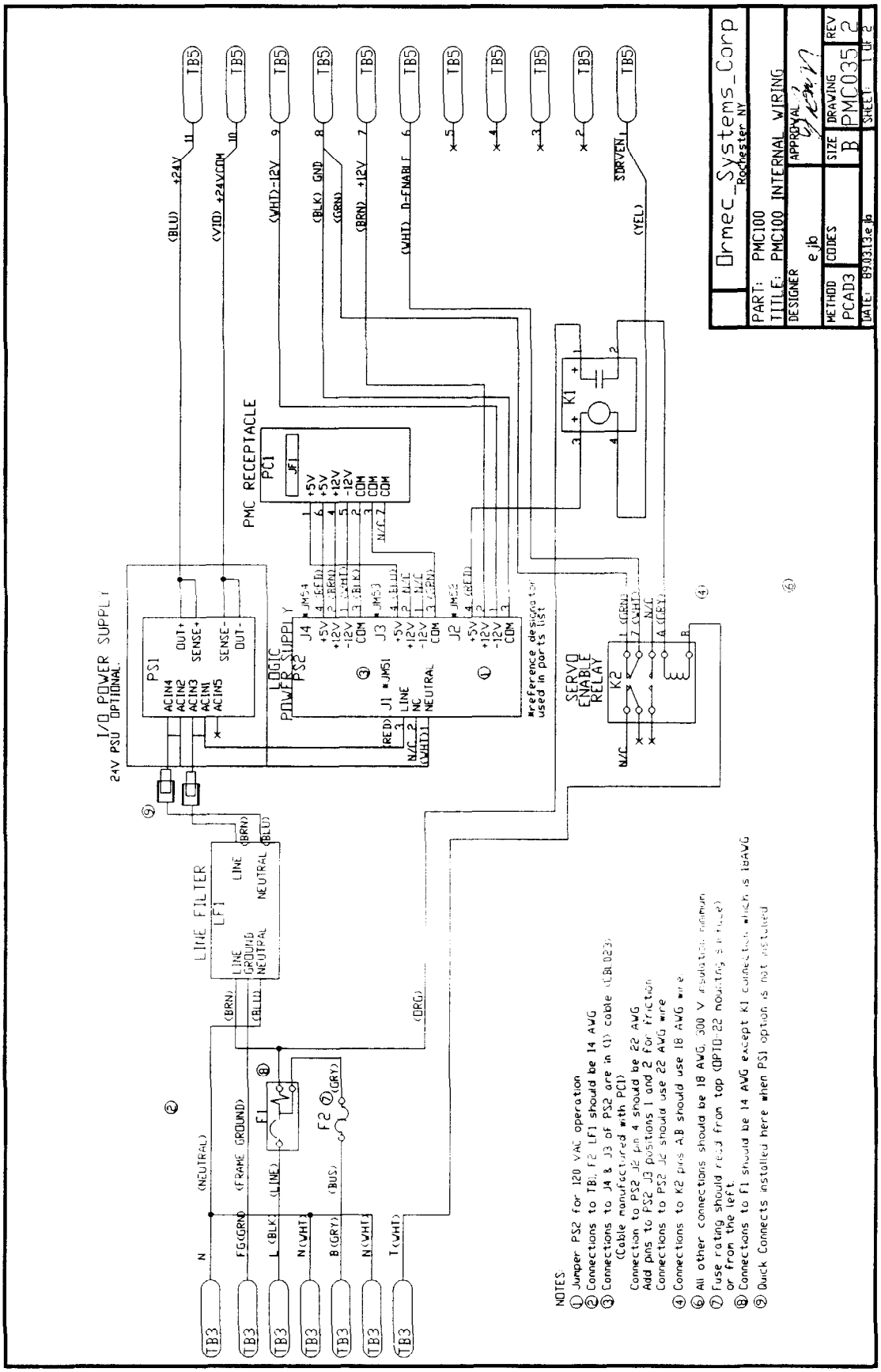

TRANSFORMER
AC DC BUS
57 75
74 100
91 125
109 150
THERMAL
COM 57V 74V 91V 109V 120V
115 VAC
TB3 T N B N L FG N
PMC-100
CB02
JM1
JF1
FAN
RECTIFIER
SERVODRIVE
CAPACITOR
RESISTOR
J4
DC BUS SUPPLY
TB2
1 EXREF
2 EXREF
3 COMMON
4 +12 VDC
5 -12 VDC
6 +5 VDC
7 SHIELD
8 N.C.
9 STOP
10 RESET
TB1
RED 1 ICMD
BLK 2 COMMON
BLU 3 SDRVEN
4 SDRVEN
5 SHIELD
6 N.C.
7 HVTACH
WHT 8 LVTACH
BLK 9 COMMON
RED 10 +5 VDC
BLU 11 ENCA
BLK 12 ENCA
YEL 13 ENCB
BLK 14 ENCB
GRN 15 ENCR
BLK 16 ENCR
17 SHIELD
18 SENSIN
19 SENSIN
20 EXVCMD
21 EXVELREF
MACHINE I/O
TB4
OUT1 (24)
OUT2 (25)
OUT4 (26)
OUT8 (27)
IN1 (24)
IN2 (25)
IN4 (26)
IN8 (27)
IN10 (28)
IN20 (29)
IN40 (30)
IN80 (31)
SEL
EXECUTE
READY
STOP
MOTOR2
MOTOR1
CHOKE
J3
DRIVE FAULT LED
J1
TACH(1)
TACH COMM(2)
+5 VDC(3)
COMM(4)
ENCA(7)
ENCA(8)
ENCB(11)
ENCB(12)
ENCR(9)
ENCR(13)
SHIELD(5)
MOTOR2(14)
MOTOR1(6)
FG(10)
COMMON(1)
ICMD (2)
R-ENABLE(6)
F-ENABLE(7)
D-ENABLE(8)
GND (11)
D-RESET(13)
CUR-MON(15)
JF2
SERVODRIVE RESET
MTE
CB001
SDRVEN CUR-MON D-RESET R-ENABLE F-ENABLE D-ENABLE +12VDC GND -12VDC +24 COM +24 VDC
TB5 1 2 3 4 5 6 7 8 9 10 11
ORMEC SYSTEMS CORP
PART NO.: MCS-920
TITLE: FUNCTIONAL WIRING DIAGRAM
DATE 061389
DESIGNER JWB
DRAWN BY
APPROVR
SHEET BMCS016-6
METHOD ACAD
REV OF 2 11

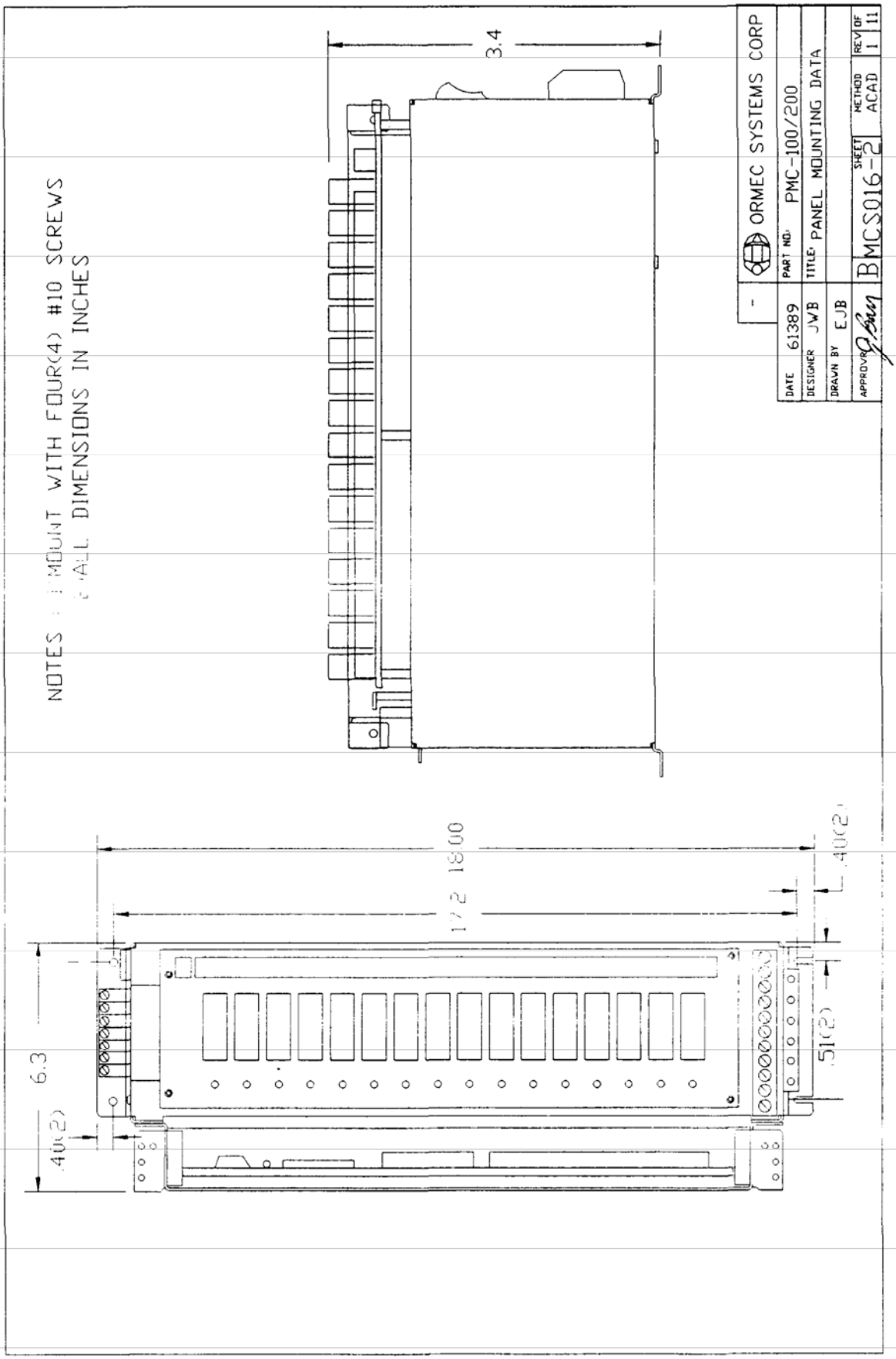

95
NOTES:
1 MOUNTING SCREWS (4) #10
2 ALL DIMENSIONS ARE IN INCHES
ORMEC SYSTEMS CORP
PART NO: MCS-920
TITLE: PANEL MOUNTING DATA
DATE 61389
DESIGNER JWB
DRAWN BY EJB
APPROVR
SHEET BMCS016-7
METHOD ACAD
REV 1
OF 11
9.75
ORMEC
DRIVE FAULT
PEAK CURRENT LIMIT
SERVO DRIVE RESET

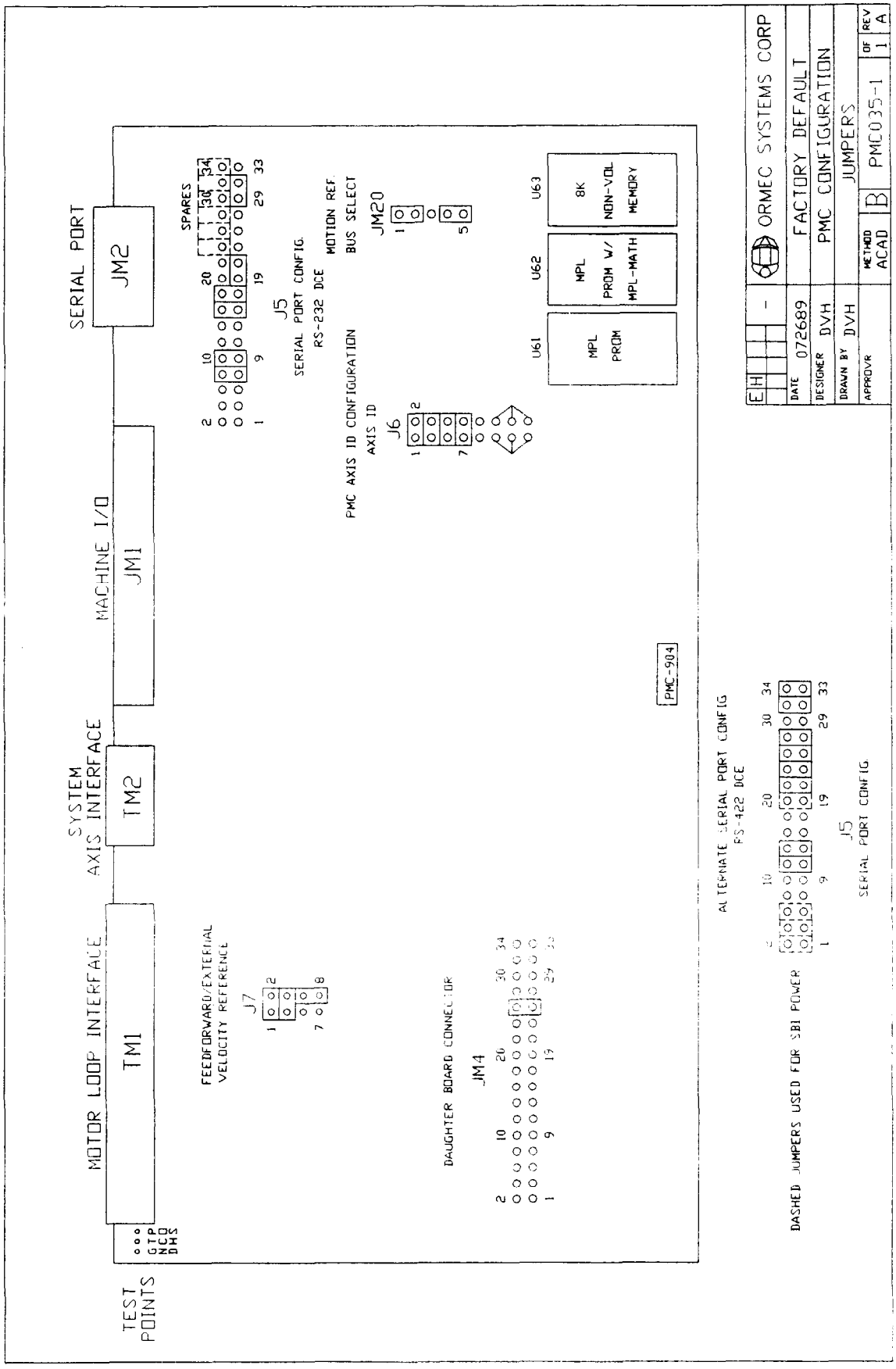

ORMEC SYSTEMS CORP
Servodrive Main Circuit
Board Layout Diagram
DATE 08-25-88
DRAWN
APPRVD
B M,C,S,0,1,6
REV 1
SHT 8
OF 11
TEST POINTS
TP1-TP4
J1
J2
J3
J4
J5
FAULT LED
PEAK CURRENT LIMIT
R28
BAL
R23
SIG
R5
A1
A3
R4
R36

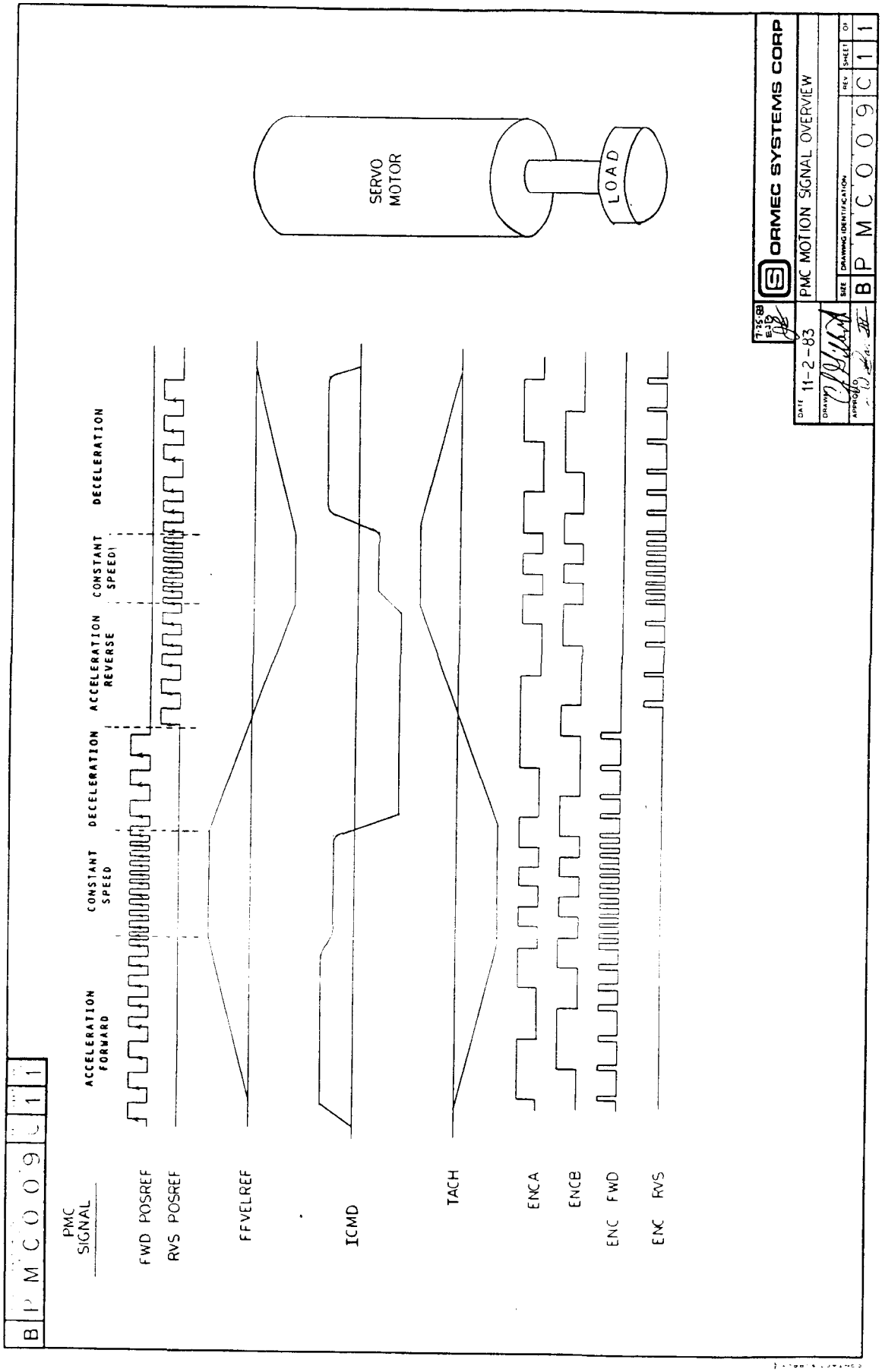

NOTES: Test Points TP1 & TP4 are for factory use only.
RESET PB is mounted on baseplate. Wiring shown here.
Switch is in parallel to TB5 D-RESET connection.
Ormec Systems Corp
Rochester NY
PART: MCS-920__PWM Servodrive
TITLE: Functional Diagram
DESIGNER J. Barr
APPROVAL
METHOD PCAD2
CODES
SIZE B
DRAWING MCS016
REV 1
DATE: 8/06/13.pol
SHEET: 1 OF 1
J4
+ BUS
MOTOR
MOTOR
- BUS
DRIVERS & H BRIDGE
PEAK CURRENT LIMIT
R28
RMS CURRENT LIMIT
R36
CONTROL LOGIC
J3
+12V
-12V
TP3
J2
DRIVE FAULT
A1A FAULT DETECTED
R52
R73
A3B
R71
R49
R46
R47
R50
TP2
SIGNAL GAIN R5
A3A
R42
R43
R44
R45
R60
BALANCE
R23
R61
R53
SERVO DRIVE ENABLE
REV. TORQUE ENABLE
FOR TORQUE ENABLE
A1B
R38
R37
R17
SERVO DRIVE RESET
A2A
R21
R22
R59
CURRENT MONITOR
J5
R1
R2
J1
GND
ICMD
D-ENABLE
R-ENABLE
F-ENABLE
D-RESET
SERVO DRIVE RESET (SEE NOTE)
GND
CUR MON

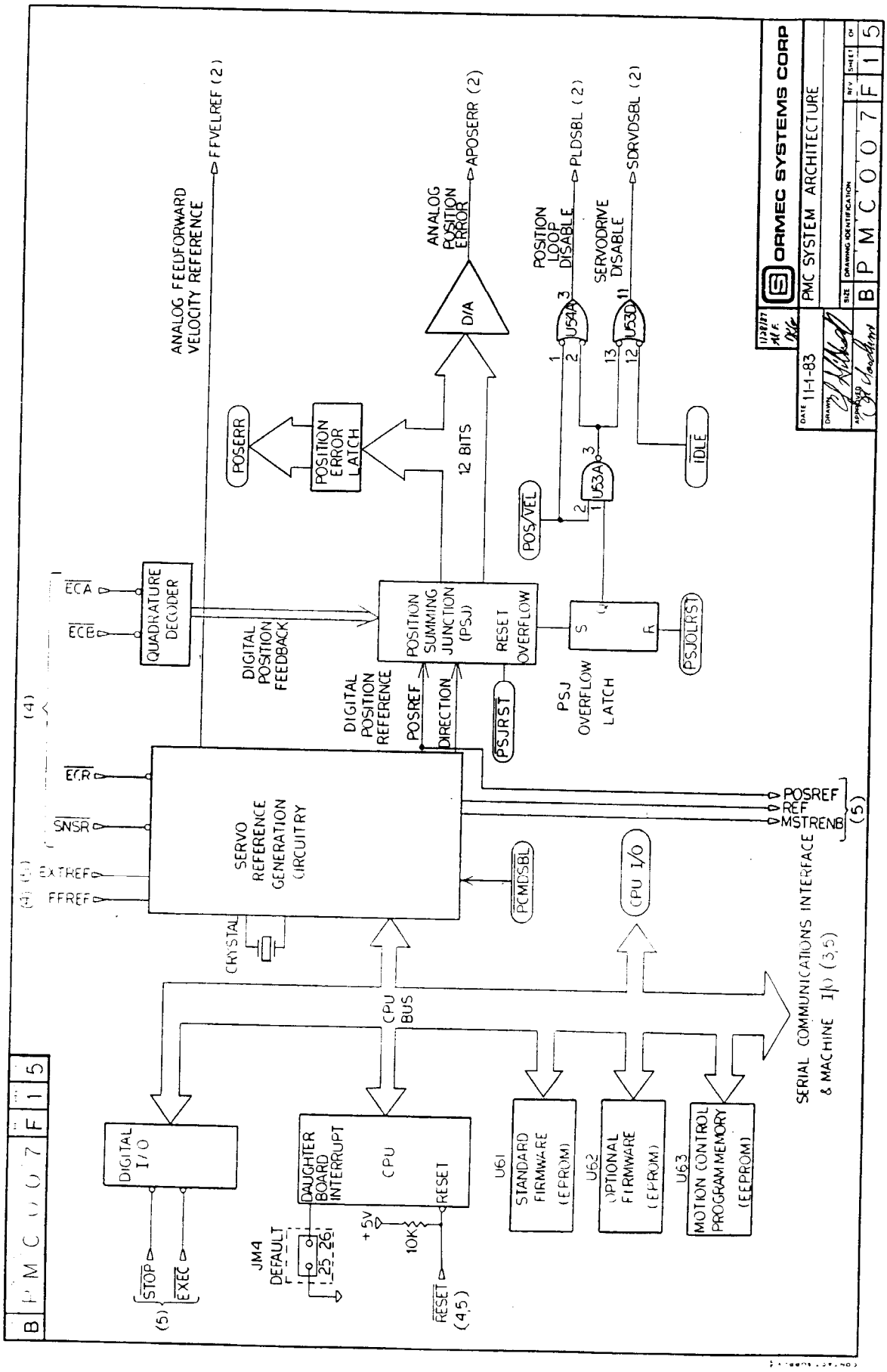

115 VAC
TB3
T N B N L FG N
PMC-100
CB02
JM1
JF1
MACHINE I/O TB4
OUT1 (24)
OUT2 (25)
OUT4 (26)
OUT8 (27)
IN1 (24)
IN2 (25)
IN4 (26)
IN8 (27)
IN10 (28)
IN20 (29)
IN40 (30)
IN80 (31)
SEL
EXECUTE
READY
STOP
TB2
01 EXREF
02 EXREF
03 COMMON
04 +12 VDC
05 −12 VDC
06 +5 VDC
07 SHIELD
08 N.C.
09 STOP
010 RESET
TB1
01 ICMD
02 COMMON
03 SDRVEN
04 SDRVEN
05 SHIELD
06 N.C.
07 HVTACH
08 LVTACH
09 COMMON
010 +5 VDC
011 ENCA
012 ENCA
013 ENCB
014 ENCB
015 ENCR
016 ENCR
017 SHIELD
018 SENSIN
019 SENSIN
020 EXVCMD
021 EXVELREF
+5 VDC
SDRVEN
TB5
CUR-MON
D-RESET
R-ENABLE
F-ENABLE
D-ENABLE
+12VDC
GND
-12VDC
+24 COM
+24 VDC
ORMEC SYSTEMS CORP
PART NO.: MCS-920
TITLE: PMC-100 WIRING DIAGRAM
DATE 082488
DESIGNER JWB
DRAWN BY
APPROVR
SHEET BMCS016-3
METHOD ACAD
REV 1
OF 11

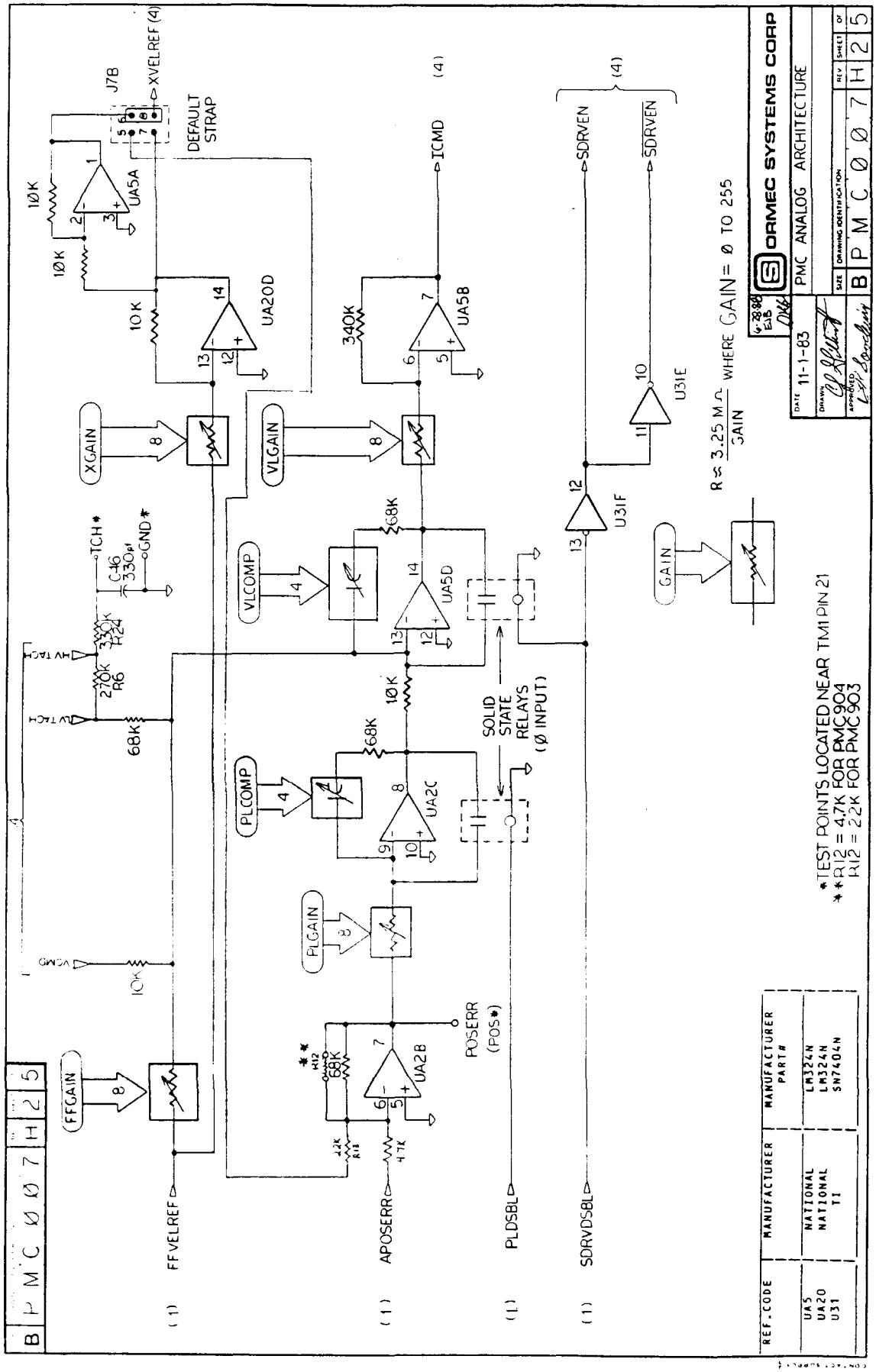

Ormec Systems Corp
Rochester NY
PART: PMC100
TITLE: PMC100 INTERNAL WIRING
DESIGNER e.jb
APPROVAL
METHOD PCAD3
CODES
SIZE B
DRAWING PMC035
REV 2
DATE: 8/30/13 e.jb
SHEET 1 OF 2
I/O POWER SUPPLY
24V PSU OPTIONAL
PS1
LOGIC POWER SUPPLY PS2
PMC RECEPTACLE
PC1
J1
J2
J3
J4
LINE FILTER
LF1
F1
F2
K1
K2
SERVO ENABLE RELAY
TB5
TB3
(BLU) +24V
(VIO) +24VCOM
(WHT) -12V
(BLK) GND
(GRN)
(BRN) +12V
(WHT) D-ENABLE
(YEL)
(BRN)
(BLU)
(ORG)
(NEUTRAL)
(FRAME GROUND)
(LINE)
(BUS)
(GRY)
N
FG(GRN)
L (BLK)
N(WHT)
B(GRY)
T(WHT)
NOTES
① Jumper PS2 for 120 VAC operation
② Connections to TB1, F2 LF1 should be 14 AWG
③ Connections to J4 & J3 of PS2 are in (1) cable (CBL023)
(Cable manufactured with PC1)
Connection to PS2 J2 pin 4 should be 22 AWG
Add pins to PS2 J3 positions 1 and 2 for friction
Connections to PS2 J2 should use 22 AWG wire
④ Connections to K2 pins A,B should use 18 AWG wire
⑤ All other connections should be 18 AWG, 300 V insulation minimum
⑦ Fuse rating should read from top (OPTIO 22 mounting style)
or from the left
⑧ Connections to F1 should be 14 AWG except K1 connection which is 18AWG
⑨ Quick Connects installed here when PS1 option is not installed

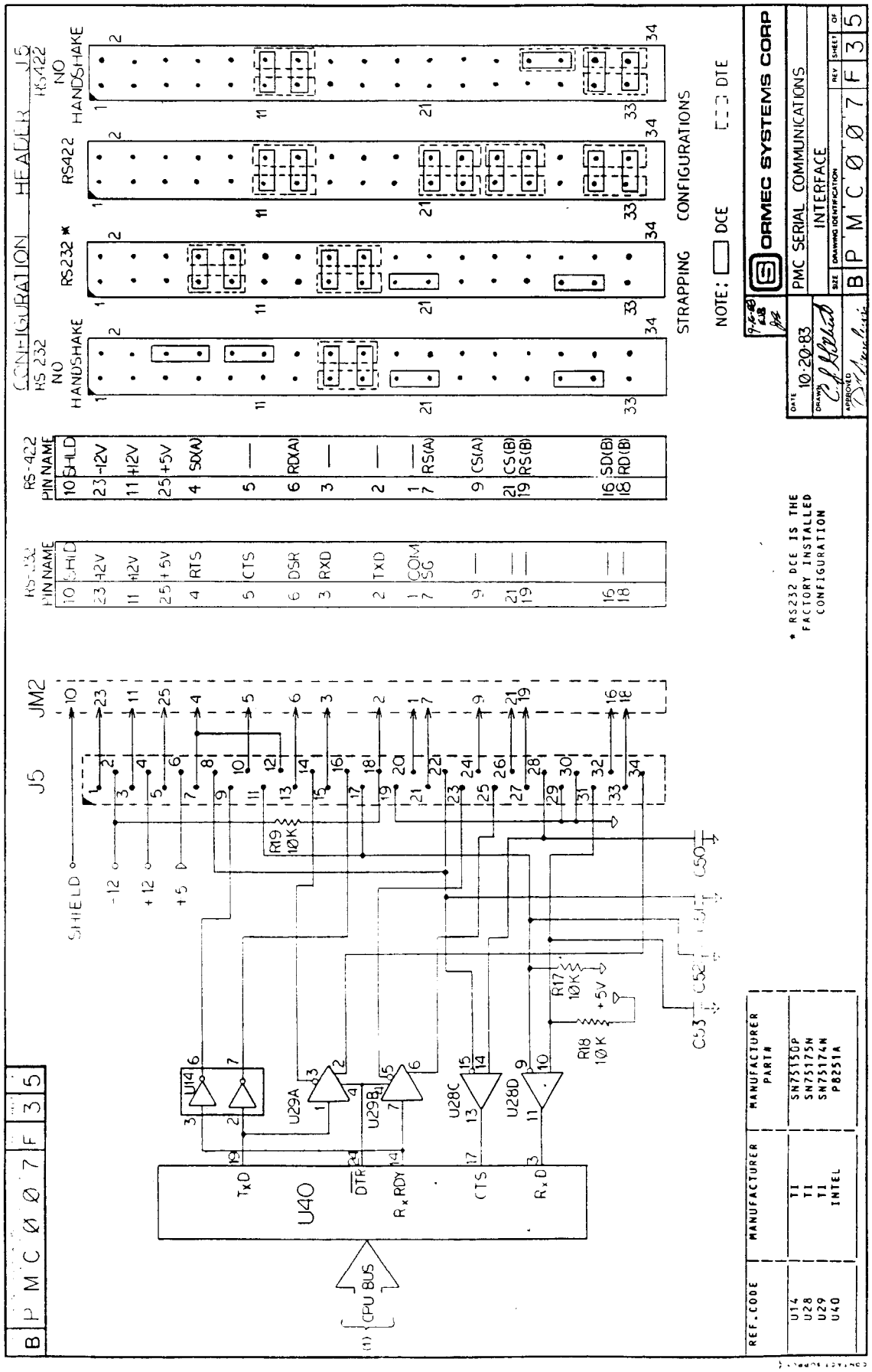

NOTES : 1.MOUNT WITH FOUR(4) #10 SCREWS
2.ALL DIMENSIONS IN INCHES
3.4
ORMEC SYSTEMS CORP
PART NO: PMC-100/200
TITLE: PANEL MOUNTING DATA
DATE 61389
DESIGNER JWB
DRAWN BY EJB
APPROVED
BMCS016-2
SHEET 1 OF 1
METHOD ACAD
REV 11
1.72
1.800
.40(2)
.51(2)
6.3
.40(2)

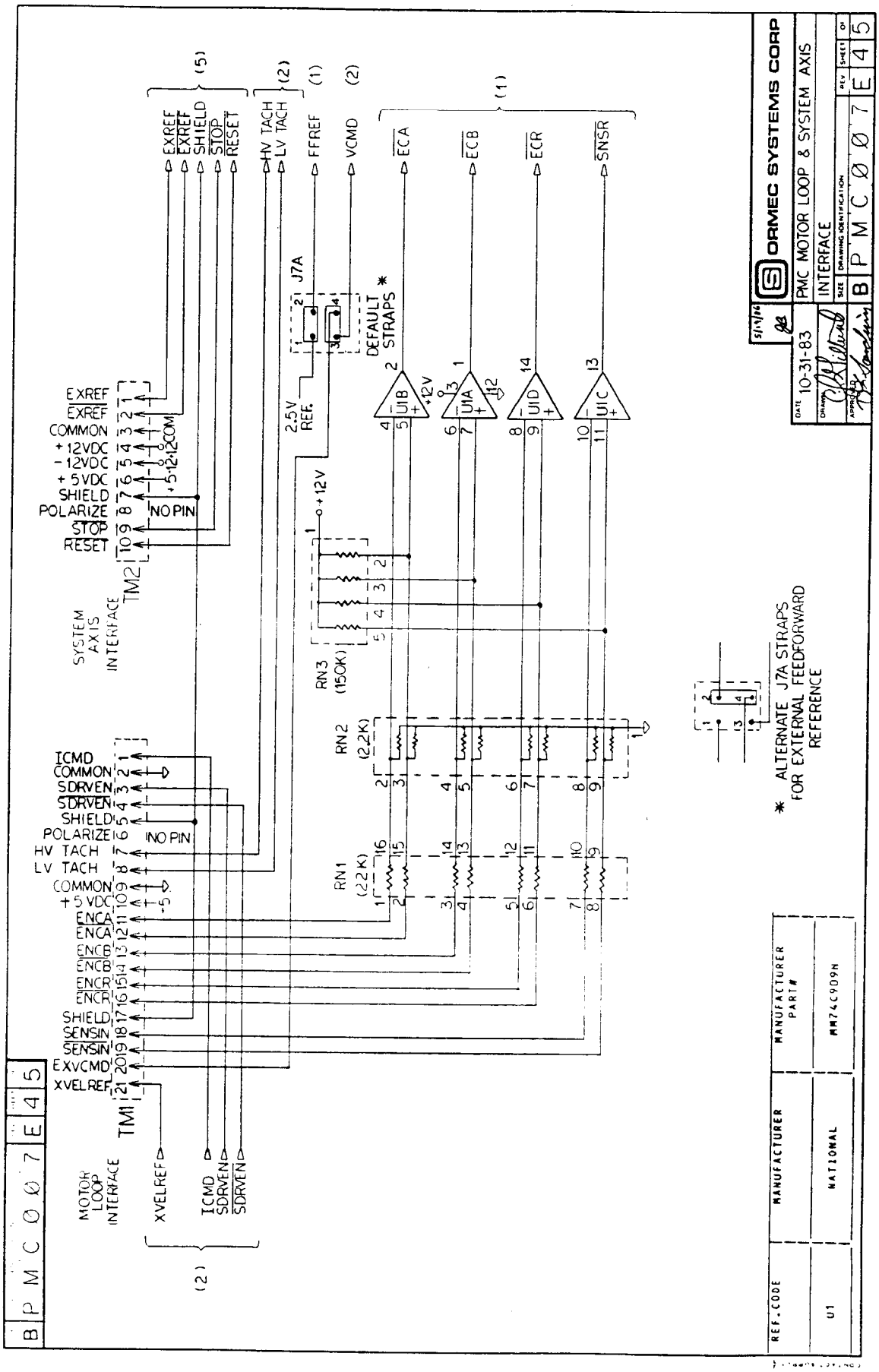

TEST POINTS
GTP
NCD
DHS
MOTOR LOOP INTERFACE
TM1
SYSTEM AXIS INTERFACE
TM2
MACHINE I/O
JM1
SERIAL PORT
JM2
FEEDFORWARD/EXTERNAL VELOCITY REFERENCE
J7
DAUGHTER BOARD CONNECTOR
JM4
PMC AXIS ID CONFIGURATION
AXIS ID
J6
SERIAL PORT CONFIG.
RS-232 DCE
J5
SPARES
MOTION REF
BUS SELECT
JM20
U61
MPL PROM
U62
MPL PROM W/ MPL-MATH
U63
8K NON-VOL MEMORY
PMC-904
ALTERNATE SERIAL PORT CONFIG
RS-422 DCE
J5
SERIAL PORT CONFIG.
DASHED JUMPERS USED FOR SBI POWER
ORMEC SYSTEMS CORP
FACTORY DEFAULT
PMC CONFIGURATION
JUMPERS
DATE 072689
DESIGNER DVH
DRAWN BY DVH
APPROVR
METHOD ACAD
PMC035-1
REV A

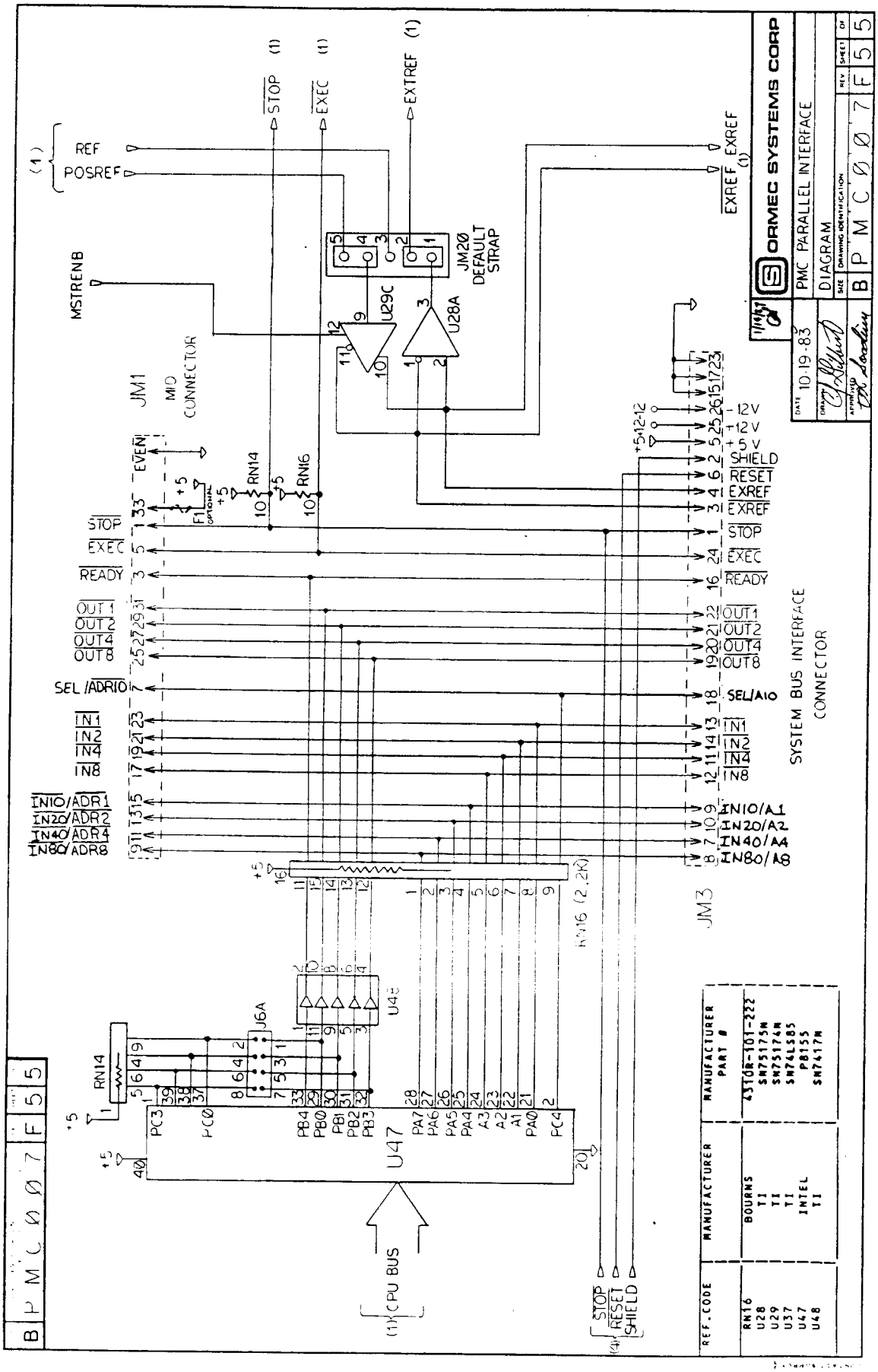

B P M C 0 0 9 C 1 1
PMC SIGNAL
ACCELERATION FORWARD
CONSTANT SPEED
DECELERATION
ACCELERATION REVERSE
CONSTANT SPEED
DECELERATION
FWD POSREF
RVS POSREF
FFVELREF
ICMD
TACH
ENCA
ENCB
ENC FWD
ENC RVS
SERVO MOTOR
LOAD
ORMEC SYSTEMS CORP
PMC MOTION SIGNAL OVERVIEW
DATE 11-2-83
DRAWN
APPROVED
SIZE B
DRAWING IDENTIFICATION P M C 0 0 9
REV C
SHEET 1 OF 1


ORMEC SYSTEMS CORP
PMC SYSTEM ARCHITECTURE
B PMC007 F 1 5
DATE 11-1-83
ANALOG FEEDFORWARD VELOCITY REFERENCE
FFVELREF (2)
ANALOG POSITION ERROR
APOSERR (2)
POSITION LOOP DISABLE
PLDSBL (2)
SERVODRIVE DISABLE
SDRVDSBL (2)
D/A
12 BITS
POSERR
POSITION ERROR LATCH
IDLE
POS/VEL
U54A
U53A
U53B
ECA
ECB
QUADRATURE DECODER
DIGITAL POSITION FEEDBACK
POSITION SUMMING JUNCTION (PSJ)
RESET OVERFLOW
PSJOLRST
PSJ OVERFLOW LATCH
DIGITAL POSITION REFERENCE
POSREF
DIRECTION
PSJRST
ECR
SNSR
EXTREF
FFREF
SERVO REFERENCE GENERATION CIRCUITRY
POSREF
REF
MSTRENB
PCMDSBL
CPU I/O
CRYSTAL
CPU BUS
SERIAL COMMUNICATIONS INTERFACE & MACHINE I/O (3,5)
DIGITAL I/O
STOP
EXEC
DAUGHTER BOARD INTERRUPT
CPU
RESET
+5V
10K
JM4 DEFAULT
RESET (4,5)
U61 STANDARD FIRMWARE (EEPROM)
U62 OPTIONAL FIRMWARE (EEPROM)
U63 MOTION CONTROL PROGRAM MEMORY (EEPROM)


ORMEC SYSTEMS CORP
PMC ANALOG ARCHITECTURE
B PMC ØØ7 H 2 5
DATE 11-1-83
DRAWN
APPROVED
J7B
DEFAULT STRAP
XVELREF (4)
UA5A
UA20D
UA5B
UA5D
UA2C
UA2B
XGAIN
VLGAIN
VLCOMP
PLCOMP
PLGAIN
FFGAIN
GAIN
ICMD
SDRVEN
SDRVEN
U31E
U31F
SOLID STATE RELAYS (Ø INPUT)
POSERR (POS*)
FFVELREF
APOSERR
PLDSBL
SDRVDSBL
R ≤ 3.25 MΩ / GAIN WHERE GAIN = Ø TO 255
*TEST POINTS LOCATED NEAR TM1 PIN 21
**R12 = 4.7K FOR PMC904
R12 = 22K FOR PMC903
REF. CODE
MANUFACTURER
MANUFACTURER PART#
UA5 UA20 U31
NATIONAL NATIONAL TI
LM324N LM324N SN7404N


CONFIGURATION HEADER J5
RS 232 NO HANDSHAKE
RS232 *
RS422
RS422 NO HANDSHAKE
STRAPPING CONFIGURATIONS
NOTE: DCE DTE
RS-422 PIN NAME
RS-232 PIN NAME
10 SHLD
23 +12V
11 +12V
25 +5V
4 SD(A)
5 —
6 RD(A)
3 —
2 —
1 —
7 RS(A)
9 CS(A)
21 CS(B)
19 RS(B)
16 SD(B)
18 RD(B)
4 RTS
5 CTS
6 DSR
3 RXD
2 TXD
1 COM SG
* RS232 DCE IS THE FACTORY INSTALLED CONFIGURATION
ORMEC SYSTEMS CORP
PMC SERIAL COMMUNICATIONS INTERFACE
DRAWING IDENTIFICATION
B P M C Ø Ø 7
SHEET 3 OF 5
DATE 10-20-83
J5 JM2
SHIELD
-12
+12
+5
R19 10K
R17 10K
R18 10K
+5V
U40
CPU BUS
TxD
DTR
RxRDY
CTS
RxD
U14
U29A
U29B
U28C
U28D
C53
C50
REF. CODE
MANUFACTURER
MANUFACTURER PART #
U14 TI SN75150P
U28 TI SN75175N
U29 TI SN75174N
U40 INTEL P8251A


ORMEC SYSTEMS CORP
PMC MOTOR LOOP & SYSTEM AXIS INTERFACE
DATE 10-31-83
SIZE B
DRAWING IDENTIFICATION PMC007
REV
SHEET 1 OF 5
B PMC007 E 4 5
MOTOR LOOP INTERFACE
TM1
XVELREF
ICMD
SDRVEN
SDRVEN
EXVCMD
XVELREF
SHIELD
SENSIN
SENSIN
ENCR
ENCR
ENCB
ENCB
ENCA
ENCA
+5VDC
COMMON
HV TACH
LV TACH
POLARIZE
SHIELD
SDRVEN
SDRVEN
COMMON
ICMD
SYSTEM AXIS INTERFACE
TM2
RESET
STOP
SHIELD
EXREF
EXREF
+5VDC
-12VDC
+12VDC
COMMON
POLARIZE
SHIELD
STOP
RESET
EXREF
EXREF
HV TACH
LV TACH
FFREF
VCMD
ECA
ECB
ECR
SNSR
J7A
DEFAULT STRAPS *
2.5V REF.
+12V
RN1 (22K)
RN2 (22K)
RN3 (150K)
UIA
UIB
UIC
UID
* ALTERNATE J7A STRAPS FOR EXTERNAL FEEDFORWARD REFERENCE
REF. CODE
MANUFACTURER
MANUFACTURER PART#
U1
NATIONAL
MM74C909N


ORMEC SYSTEMS CORP
PMC PARALLEL INTERFACE
DIAGRAM
B PMC0007 F 5 5
DATE 10-19-83
SYSTEM BUS INTERFACE CONNECTOR
JM1 MID CONNECTOR
JM3
U47
U48
U28A
U29C
JM20 DEFAULT STRAP
J6A
RN14
RN16
RN16 (2.2K)
CPU BUS
REF
POSREF
MSTRENB
STOP
EXEC
EXTREF
EXREF
EVEN
READY
RESET
SHIELD
SEL/ADR10
SEL/A10
OUT1
OUT2
OUT4
OUT8
IN1
IN2
IN4
IN8
IN10/ADR1
IN20/ADR2
IN40/ADR4
IN80/ADR8
IN10/A1
IN20/A2
IN40/A4
IN80/A8
+5
+12V
-12V
REF. CODE
MANUFACTURER
MANUFACTURER PART #
RN16 BOURNS 4310R-101-222
U28 TI SN75175N
U29 TI SN75174N
U37 TI SN74LS85
U47 INTEL P8155
U48 TI SN7417N