

BALDOR
MOTORS AND DRIVES

PROGRAMMABLE POSITIONING CONTROLLER

SMCC
SMART MOTION
CONTROL CARD

INSTALLATION AND OPERATING MANUAL

Table of Contents

Section 1	
Overview	1-1
Section 2	
Specifications	
2.1 Physical Specifications	2-1
2.2 Power Requirements	2-1
2.3 RS232 Port	2-1
2.4 Parallel Port	2-1
2.5 Memory Capacity	2-1
2.6 Servo Performance	2-1
2.7 Velocity Control	2-2
2.8 Position Control	2-2
2.9 Encoder Interface	2-2
2.10 Handwheel Encoder	2-2
2.11 Inputs/Outputs To Amplifier Driver	2-3
2.12 Analog Outputs	2-3
2.13 Analog Inputs	2-3
2.14 Digital Outputs	2-3
2.15 Manual Control Inputs	2-4
Section 3	
Getting Started	
3.1 Setting Up the Hardware	3-1
3.1.1 Motor/Control Connections	3-1
3.1.2 Setting up for Host Communication	3-1
3.1.3 Power-up and Communication Diagnostics	3-3
3.2 Communication and Command Format	3-4
3.2.1 Command/Instruction Format	3-4
3.2.2 Card Address Select Commands	3-5
3.3 Set Up Initialization Parameters	3-6
3.3.1 Motor Type Selection	3-6
3.3.2 Encoder Setup	3-7
3.3.3 System Reference Speed Setup	3-7
3.3.4 System Limit Setup	3-8
3.3.5 Parameter For Special Applications	3-9
3.3.6 Operational Diagnostics and System Tuning	3-10
3.3.7 Use of External Pulse (Handwheel) Input	3-13
3.3.8 Use of Analog Command Input	3-13
3.3.9 Position/Velocity Display Setup	3-14
3.3.10 Home Setup	3-15
3.3.11 PLC Enable/Disable	3-15
3.4 Simple Operations	3-16

Section 4

Direct Commands

4.1 Motion Commands	4-1
4.2 Program Control Commands	4-2
4.3 Program Editing Commands	4-6
4.4 Status Query Commands	4-8
4.5 Control Character Commands	4-11
4.6 Hands-On Command Examples	4-12

Section 5

Buffer Instructions

5.1 Program Definition/Control Instructions	5-1
5.2 Motion Definition Instructions	5-9
5.3 Motion Instructions	5-13
5.4 Hands-On Application Programs	5-16

Section 6

PLC Programming

6.1 Direct PLC Commands	6-1
6.2 PLC Buffer Instructions	6-1
6.3 PLC Program Examples	6-6

Section 7

Application Programs

Examples

7.1 Single Axis Torch Cutter	7-1
7.2 Two DC Motors in a Pick and Place Operation	7-2
7.3 Serpentine Pattern	7-5
7.4 Thumbwheel Switch Application	7-6
7.4.1 Thumbwheel Switch Connection Without Addition Circuitry	7-7
7.4.2 Thumbwheel Switch Connection With Additional Circuitry	7-8
7.4.3 Speed (RPM) Control Using Thumbwheel Switches	7-9
7.4.4 Speed and Job Number Control Using Thumbwheel Switches	7-11
7.5 Glue Dispenser Program	7-16
7.6 Altered Destination On Trigger	7-18
7.7 Multiple SMCC Sequencing Method	7-20
7.8 An Automated Encoder Line Counting Program	7-24
7.9 "S" Curve Accel and Decel (Requires Parabolic Option)	7-25

Section 8

Technical Information

8.1 Parabolic Profiling	8-1
8.1.1 Direct Specification	8-1
8.1.2 S-Curve (three-point Smoother or CIRCLE 4 MODE)	8-2
8.1.3 General Algorithm for point-to-point S-Curve Moves	8-4
8.2 1/T Encoder Period	8-8

8.3	Data Gathering	8-8
8.4	DMA Communication	8-10
Appendix A		
Glossary of Initialization/Setup Commands		A-1
Appendix B		
SMCC Status Codes		
B.1	SMCC On-Board LED Display Codes	B-1
B.2	SMCC Status Bytes	B-1
Appendix C		
On-Board DIP Switch		
C.1	Card Address Selection Switch	C-1
C.2	EAROM Enable Switch	C-1
C.3	Communication Mode and Baud Rate Selection Switch	C-2
Appendix D		
SMCC Memory and I/O Map		D-1
Appendix E		
SMCC E-Points and Test Points		E-1
Appendix F		
SMCC Connectors and Pin Assignments		F-1
Appendix G		
Summary of SMCC Commands		
G.1	Direct Commands	G-1
G.1.1	Card Address Select Command	G-1
G.1.2	Direct Motion Commands	G-1
G.1.3	System Control Commands	G-1
G.1.4	Status Query Commands	G-2
G.1.5	Editing Commands	G-2
G.1.6	Advanced Command Set	G-3
G.1.7	Control Key Commands	G-3
G.2	Buffer Instructions	G-3
G.2.1	Program Flow Control Commands	G-3
G.2.2	Motion Control Commands	G-4
G.2.3	Advanced Instruction Set	G-5
G.3	PLC Commands/Instructions	G-5
Summary of Parabolic SMCC i-Parameters		
	For Brush Type DC Motor Controllers	G-6
	For Brushless AC Motor Controllers	G-8
Index		I-1

—

—

—

Section 1

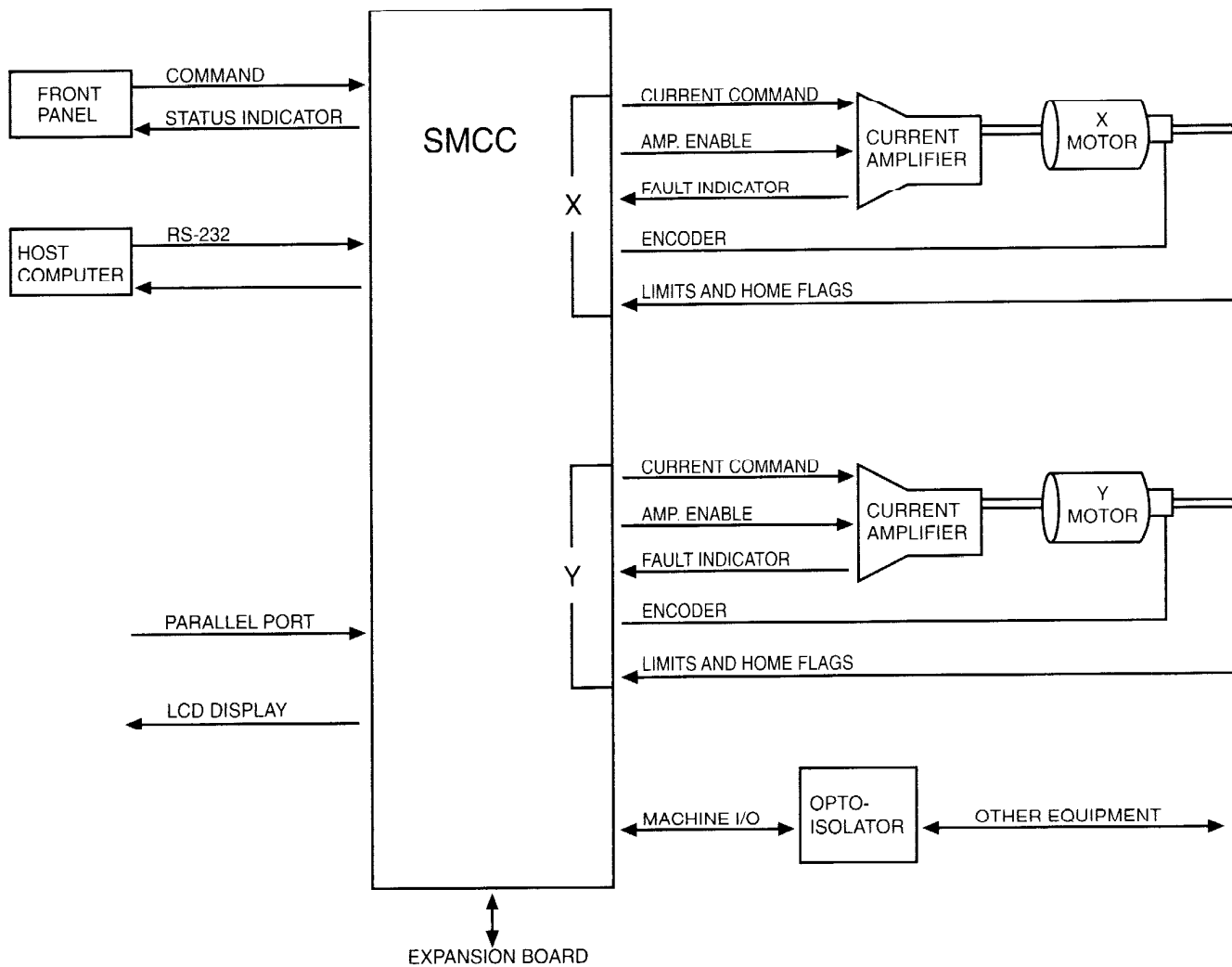
Overview

GENERAL

The Smart Motion Control Card (SMCC) is a 2axis digital motion controller with built-in programming capability. It has the ability to control two DC or AC motors with the appropriate controls. When multiple SMCCs are connected in daisy chain, synchronized motions for up to 16 SMCC cards (32 axes) is possible. Refer to the functional block diagram on the next page.

FEATURES:

- Fast 16bit pipelined custom microprocessor control.
- Compact size, fits in control chassis or in stand-alone enclosure.
- Host communication via RS232 serial port, or high speed bidirectional parallel port.
- Precise synchronization of linear or circular interpolation for up to 32 axes.
- Trapezoidal or smooth “S” curve velocity profile. Cubic path generator for precise curve fitting.
- Fast servo update rate up to 2 kHz.
- Digital error integration loop to reduce following error and control drift.
- Digital PID control with velocity and acceleration feedforward gain.
- Feedback signal incremental encoder or Pulse and Direction input.
- 1/T period encoder measurement for accurate low speed control
- Feedrate and Time mode programming. In time mode, all moves can be executed synchronously within a specified time.
- Data (Position and Following Error) capturing capability. Can be plotted with “Monitor” program.
- Provision for an LCD Display to indicate position, velocity, and operating status.
- Structured English-like commands and program instructions.
- Accepts external handwheel and thumbwheel switch control.
- On-board PLC (Programmable Logic Control) capability.
- Can perform electronic commutation of Brushless or Induction motors with appropriate controls.



TYPICAL APPLICATION OF SMCC

Section 2

Specifications

2.1 Physical Specifications

- Size: Standard Double-Euro size (9.2 x 6.3 x 0.75 in.)
- Weight: 0.3 lbs. without mounting plate; 1.2 lbs. with mounting plate
- Operating Temperature: 32 to 140 F.
- Storage Temperature: 10 to 180 F
- Humidity: 0 to 90%, non-condensing

2.2 Power Requirements

- +5VDC +/-5%, 1.0A with all options and two encoders
- +/-15VDC (or +/-12VDC) for optically-coupled control command output (Usually provided by the control). Current requirements are 50mA on the positive supply and 15mA on the negative supply.
- +/-15VDC, 100 mA is required if an optional D/A, A/D, or R/D converter is installed.

2.3 RS232 Port

- Baud rate selectable by DIP switch (300 to 38,400 Baud)
- Line parameters: 8data, 1 stop, No parity, half duplex.
- Via daisy-chained operation, the host computer can address, and control multiple SMCCs.

2.4 Parallel Port

- An alternative host communication port: 8bit parallel data transfer, with handshake; The average per character transfer time is 120 microseconds. This can be used for binary DMA communication at 3 to 4 microseconds/character data rates.
- With programming, it can be used to accept thumbwheel switch inputs (up to 16 channels without external decoding) or used as 8 additional TTL Outputs.

2.5 Memory Capacity

- 32K RAM and 8K EEPROM standard; can store up to 5,630 steps of program.

2.6 Servo Performance

- 480 uSec. per axis servo update time.
- Execution rate is 100 blocks/Sec or 10 mSec/block when controlling one axis and 60 blocks/Sec when controlling two axes.

2.7 Velocity Control

- 0.005 to 1,000,000 counts/Sec. Employs crystal-controlled digital phase locked loop, with the RPM range dependent on the encoder's resolution.
- 1/T encoder measurement provides enhanced smoothness at low speeds.
- Velocity Accuracy:
Long Term: 0.002% absolute
Short Term: System-dependent, typically 0.5 to 1%
Multi-Axis: 0.001%, adjustable to zero following error between axes
- Repeatability: 0.001%

2.8 Position Control

- Position Range: +/-67 million encoder counts
- Position Accuracy: +/-one encoder count

2.9 Encoder Interface

- Independent X and Y channels for position control
- A/B quadrature with "C" (Home) channel
- Single-ended or differential interface selectable
- X1, X2 and X4 multipliers for A/B quadrature mode and selectable pulse direction mode
- Maximum input rate: 500,000 counts/sec with X1 multiplier or in the pulse/direction mode; 1,000,000 counts/sec with X2 multiplier; 800,000 counts/sec with X4 multiplier. All rates are after multiplication.

2.10 Handwheel Encoder

- Independent X and Y channels
- Quadrature encoder inputs or pulse and direction inputs (external synchronization)
- Maximum input rate: 500,000 counts/sec with X1 multiplier or the pulse/direction mode; 1,000,000 counts/sec with X2 multiplier; 1,600,000 counts/sec with X4 multiplier. All rates are after multiplication.

2.11 Inputs/Outputs To Amplifier Driver

- Optically isolated error outputs (command signal to the control), control disable and control fault input signals.
- All signals are referenced to the control common to achieve total isolation between the SMCC's digital circuits and the power section.

2.12 Analog Outputs

- When designated as two-axis control, it provides with standard $\pm 10V$ DC (torque or velocity) error outputs and 9 bits of resolution.
- Software-selectable sinusoidal, trapezoidal, or six-step phase commutation waveforms for brushless DC or AC induction motors.

2.13 Analog Inputs

- 4 analog inputs:
 - 2 are used to monitor the motor current and to provide programmable i_{2t} protection
 - 1 is used as a 0 to 100% feed rate control with a simple potentiometer input
 - 1 is used as a bipolar velocity control with range multiplication using the jog inputs.

2.14 Digital Outputs

- $\pm$ /limits for the X and Y axis
- Control disable output
- Control fault input
- X and Y home flag inputs for enabling encoder "C" channel
- Two user flag inputs for precision, interrupt-driven, detection of external events. Position capture with less than 8 microseconds error.
- Seven input channels: TTL compatible, 0 to +5V. When employed with controls equipped with error code generation, 3 of the 7 inputs may be used to display control error codes. All inputs may be read under program or PLC control.
- Five general purpose output channels: Open collector, 30V, one at 60mA (MA01), four at 200mA (MA02MA05). All outputs are programmable "on the fly" by program or PLC control.

2.15 Manual Control Inputs (Requires Control Panel)

Name	Equiv. Command	Function
XJPL/	J	X-axis, Jog Positive
XJMI/	j	X-axis, Jog Negative
YJPL/	JY	Y-axis, Jog Positive
YJMI/	jy	Y-axis, Jog Negative
PREFJ/	=	Return to pre-jog or pre-handwheel position.
STRT/	R	Starts continuous program execution.
STEP/	S	Executes one block of data at a time.
STOP/	q	Stops running immediately, resets program
HOME/	h	Starts X and Y axes on a Homing cycle.
HOLD/	H	Stops motion immediately.

CONTROL PANEL OUTPUTS	
Name	Function
IPLD/	In-Position Indicator
BFLD/	Buffer Full Indicator
ERLD/	Control Fault Indicator
EPLD/	End of Program Indicator
ITLD/	Current Limit Indicator
F1LD/	Following Error 1
F2LD/	Following Error 2

Section 3 Getting Started

3.1 SETTING UP THE HARDWARE

3.1.1 Motor/Control Connections

All wiring should be made according to the chassis manual. As a minimum, connection to encoders, controls and motors, and RS232 communication link to host computer should be done.

The wiring of the limit switches must be performed. If the limit switches are not used, disable the limit protection by grounding the four limit switch lines XPLL, XMIL, YPLL, YMIL. Consult the chassis manual for the exact location of these pins.

Make sure that all of the screw terminals and connections are securely tightened. Proper grounding guards against electrical shock to personnel and can reduce the effects of noise interference.

3.1.2 Setting up for Host Communication

An on-board 8-digit DIP switch is set such that all 8 switches are OFF (pressed to the side marked as "OPEN") which means the following default settings:

- (a) Card address is set to A0 (Switch No. 1,2,3 and 4 OFF).
- (b) 9600 baud serial communication (Switch No. 6,7,8 OFF).
- (c) Save enabled (Switch No.5 OFF). Modified parameters and program can be altered permanently by "s" command.

In addition, the following jumpers must be installed correctly:

- (d) If the card address is set as A0, it must always act as a "master" supplying a SYNC clock to the other cards. (Jumper on F2 installed, Jumper on E3 removed).
- (e) Since a direct link between the host computer and the SMCC is normally used, the SMCC is configured as a "Null-Modem" so that a standard 9 line modem cable fits without modifications. (Jumpers on 12 and 34 on both E7 and E8).

If a single SMCC is to be used without daisy-chaining, all default configurations should work properly. Connect an RS232 cable between the SMCC and the host computer, and run a communication program (such as Crosstalk, Procomm, etc.). You may obtain the "Monitor" communication program from Baldor free of charge. It operates on a PC or compatible computer, and has many convenient features for host communication.

The host computer line parameters must be set as 8-bit of data, 1 stop bit, no parity, half duplex with a baud rate the same as set by the SMCC's dip switches.

The default baud rate is 9600, Other rates can be configured through the DIP switch No. 6, 7 and 8. as in the following table.

SW6	SW7	SW8	COMMUNICATION MODE
0	0	0	Serial 9600 Baud
1	0	0	Serial 4800 Baud
0	1	0	Serial 2400 Baud
1	1	0	Serial 1200 Baud
0	0	1	Parallel Communication
1	0	1	Serial 300 Baud
0	1	1	Serial 38500 Baud
1	1	1	Serial 19200 Baud

If parallel communication is to be used, you may first setup with RS232 serial communication and then switch to parallel communication once setup is completed. When a single SMCC is used, you may jump to the next section. If multiple SMCC's are connected as a daisy-chain, follow the procedure below. You may designate each SMCC card address sequentially, and set the DIP switches according to the following table:

SW1	SW2	SW3	SW4	CARD ADDRESS
0	0	0	0	A0
1	0	0	0	A1
0	1	0	0	A2
1	1	0	0	A3
0	0	1	0	A4
1	0	1	0	A5
0	1	1	0	A6
1	1	1	0	A7
0	0	0	1	A8
1	0	0	1	A9
0	1	0	1	Aa
1	1	0	1	Ab
0	0	1	1	Ac
1	0	1	1	Ad
0	1	1	1	Ae
1	1	1	1	Af

For synchronized motion, one (and only one) SMCC card, addressed as A0, provides the SYNC clock signal to the rest of the SMCCs. Install jumpers E2 and E3 as follows.

- A0 card: Jumper E2 installed, Jumper E3 removed.
- All other cards: Jumper E2 removed, Jumper E3 installed.

3.1.3 Power-up and Communication Diagnostics

At this stage, it is assumed that all the wiring and communication setup has been completed as instructed in the previous section.

STEP 1. Turn the SMCC's power supply ON while the control bus power remains OFF. Observe the on-board 7-segment LED display. It may display one of the following patterns.

- (1) ".O" (left dot), the SMCC is "in-position" and no errors are found. Go to STEP 2.
- (2) "O." (right dot), following error. proceed to set up the correct parameters. Go to STEP 2.
- (3) ".O." (two dots), the SMCC is not operating normally. Verify that all I.C.'s are securely positioned. Check for mechanical damages from shipping and handling. If damage is found, consult the factory or distributor. Otherwise, check that the +5V supply is in between 4.75 and 5.25 volts.
- (4) LED OFF ; No power is on the SMCC. Turn off the power, check the power supply connections and check for a blown fuse and try STEP 1 again.

STEP 2. Run the communication program from the host computer, and type "p" and press the Carriage Return (CR). Since you are in half duplex mode, the "p" character should be printed on the screen. The cursor on the host computer screen should respond with a line feed and the present position of the motor(s). This is the desired response telling you that the SMCC is communicating properly. Go to the next section. Otherwise, check the following.

- (1) Set up as "full duplex". If the "p" character is not shown on the screen, change to half-duplex mode on your communication program.
- (2) Line parameter mismatch. Check if 8-data, 1-stop, no parity is set. Check that the baud rate is the same as the rate set by the SMCC's DIP switches.
- (3) The serial port of your host computer may be configured as a "null modem", i.e., signals TXD, RXD and RTS, CTS are swapped. In this case, the SMCC's jumpers E7 & E8 can be changed to adapt to the null modem port. See Appendix E for the description of the jumpers.
- (4) The host computer port is not setting the "RTS" handshake line high. The SMCC supports the hardware handshake lines RTS and CTS. The required interconnect wires between the SMCC and the host computer are as a minimum:

NAME	DB-25 CONNECTOR PIN #	
TXD	2	TRANSMIT DATA
RXD	3	RECEIVE DATA
RTS	4	REQUEST TO SEND
CTS	5	CLEAR TO SEND
GND	7	GROUND

Note that DSR and DTR signals are not supported by the SMCC but are tied together at the SMCC.

Other connections will be required if multiple cards are daisy chained.

- (5) If the communication is not consistent, refer to the directions on the Baldor Monitor software (Version 3.0 or higher) for adjusting for the CPU speed. If this does not remedy the problem, check for noise on the signal. Noise may be induced if the cable is too long, or too close to the power line etc. Proper shielding, or the reduction of baud rate may help.

3.2 COMMUNICATION AND COMMAND FORMAT

3.2.1 Command/Instruction Format

The SMCC uses printable ASCII characters to communicate with the host computer. Commands are mostly abbreviated English words to make them easy to learn and remember, but special characters (like Control H, Control X etc.) are also used.

When a command is transmitted to the SMCC through the RS232 link, the SMCC stores that command but does not process it until a CR character is received. When a line of command is processed, the SMCC sends a LF character to acknowledge the reception but does not echo back the received character. This is commonly called "Half Duplex" communication mode.

The SMCC adopts the following conventions:

1. <CR> (Hex 0D, Carriage Return) character is interpreted as an "End of Line" indicator. The SMCC prompts with a <LF> (Hex 0A, Line Feed) character when a line of command is processed (or stored in buffer). This prompt <LF> can be disabled by the i03 initialization command. (see Appendix A, i03)
2. The SPACE character (Hex 2C.) is interpreted as a separation character. It is simply neglected by the SMCC. The operator may use this character for text legibility.

Example 3.1

F 1024	Valid command to set maximum feed rate
F1024	Also valid with no space

3. Backspace (Hex 08, or Control-X) can be used to cancel the previous character, while Control-X (Hex 18) can be used to cancel all characters in the current line.
4. The maximum number of characters in a single line of commands depends on the number of cards being chained. Each card can hold 24 characters in a command line, (without counting <CR> character, or address commands) Once the address command is detected, commands are stored in the newly addressed card.
5. Each card can receive only one command line at a time. There are exceptions (e.g. Card Selection Command, Label Command).
6. When Engineering units are used as a command parameter, the leading zero must be added before the fractional numbers, i.e., 0.4536, or 0.78.

Hereafter the “command” is referred to as a series of characters issued to the SMCC so that the SMCC can respond accordingly. They are categorized as “Direct Commands” and “Buffer Instructions”. Direct Commands are processed immediately, while Buffer Instructions are stored in the SMCC’s buffer and will be processed later when a R or S command is accepted.

3.2.2 Card Address Select Commands

The Address Select command has an “A” character together with a Hex number. (0, 1, 2, .. 9, a, .. f). For example, “A8” selects Card #8, and “Ab” selects Card #11. The “AA” command selects all the cards simultaneously.

Upon Reset, the processor selects “AA” (All Cards Selected) as a default. Upon transmission of a card selection command, the corresponding card becomes active while the other cards become inactive. The card select command has the highest priority, and will be processed immediately (Without waiting for a <CR> character). Also, it can be placed anywhere in the command line. One card must be selected before the user can make an SMCC status query (“p”, “v”, “inn”).

An initialization command “inn” without any parameter value is interpreted as a parameter value query, where nn could be any number between 00 - 81. The command “ix” will respond by displaying all of the parameters.

Example 3.2

Command	Function
A0	Card A0 Enabled
i00	Query to check the Following Error Fault Limit
i00 2000	New parameter value is sent to replace old value
i002000	Same as above
I00 2000	Buffer command - will be processed at the time of program execution

3.3 SET UP INITIALIZATION PARAMETERS

There are 82 i-parameters used to configure the SMCC's operation conditions, such as servo gains, following error bands, maximum reference speed.. etc. I-parameters are loaded and tested at the factory, follow the instructions to verify the factory settings and change them if necessary. Appendix A gives a detailed description for each i-parameter.

There are three ways to change these i-parameters:

1. By direct commands "inn".
2. By program using the "Inn" instruction.
3. By using the "Monitor" (Version 3.0 or later) program.

In this section, we use the direct command "inn" to set up the initialization parameters. As a convention, a pair of brackets [] means the equivalent command for the Y axis, so i20[i40] means that the identical rule can be applied to not only the i20 (X axis) but also the i40 (Y axis).

At this point, we assume that all of the interconnections to the control, motor, encoder, etc. are properly set. As a recommendation, you may first apply power only to the SMCC and host computer, do not apply power to the controls and do not connect the motor to the load. Connect power to the control/motor assembly only when the i-parameters have been properly set.

3.3.1 Motor Type Selection

- Set i13 to 0 for DC Motors (brush type) or its equivalent. Certain controls (such as Baldor's BTS15 Brushless Control) have a built-in commutation function and the control and motor act as a simulated DC motor. In this case, configure it just like the DC motor. For other drive combinations, refer to Appendix.A, i13.
- Set i15 to 0 (Two axes) or 1 (X axis only). Note that even in a one-axis application, the limit switches for the Y axis must be grounded, otherwise the SMCC will not command any moves.

3.3.2 Encoder Setup

- Set i39[i59], set position encoder mode control bits. First, the proper choice must be made in selecting the encoder type and the encoder multiplication factor.

i39	DESCRIPTIONS
0	Pulse and Direction CW (counts up when B is high)
1	x1 Quadrature Decoding (counts up when B leads A)
2	x2 Quadrature Decoding (counts up when B leads A)
3	x4 Quadrature Decoding (counts up when B leads A)
4	Pulse and Direction CCW (counts up when B is low)
5	x1 Quadrature Decoding (counts up when A leads B)
6	x2 Quadrature Decoding (counts up when A leads B)
7	x4 Quadrature Decoding (counts up when A leads B)

If we set i39[i59] to 3 (default), it means that we are using quadrature decoding and a x4 encoder multiplication factor. Setting i39[i59] to 7 has the same effect except the direction of rotation is reversed. If the motor and encoder lines were connected as instructed by the chassis manual, values 0-3 should work properly. An i39 value of 4 to 7 may be used to reverse the direction of the encoder reading (same effect as reversing the A and B channel).

The SMCC uses count values as the principal means of representing distance or velocity. If an mm ppr encoder is attached, and an n multiplication factor is selected, the counts per revolution “cm” is defined as:

$$\text{cm} = \text{mm} \times n \text{ (count per revolution)}$$

For example, if mm = 500 lines/rev. and n = 4 (i39 = 3), then cm = 2000 cpr. This means whenever the motor turns one complete revolution, the distance is recognized as 2000 counts in the SMCC. For most applications, x4 multiplication is recommended. i39[i59] and “cm” is also used in setting i06 (Reference Velocity Set).

3.3.3 System Reference Speed Setup

- Set i06 [nn] (1000-200000), the reference (maximum) speed. This specifies the reference velocity in encoder counts per second. One motor revolution corresponds to the counter value of “cm”. For example: If a 500line encoder is configured with x4 multiplication, and you wish to set the maximum motor speed at 1200 RPM (20 rps), then set i06 to $500 \times 4 \times 20 = 40,000$ [counts]
- Set i31[i51] (Range 0-255), Jog feedrate (speed), in terms of counts per second. Jog speed is calculated by

$$\text{Jog Speed} = (\text{i31[i51]} \times \text{i06}) / 256$$

For instance, i31 128 sets the X axis jog speed to half of the reference speed, while i51 64 sets the Y axis jog speed to 1/4 of the reference speed.

- Set i08 (100 50000) for an Acceleration/Deceleration time roughly in mSec (actually 1/1024 sec.). For example, i08 2048 sets the acceleration and/or deceleration time to be two seconds.

The acceleration and deceleration time must be set to a value that the control and motor system can handle. Too small of a value causes the control to be unable to react in the specified time (control overcurrent limit may occur) or may result in too much overshoot which cannot be corrected by any SMCC servo parameters.

3.3.4 System Limit Setup

- Set i00 2000 (default) for following error fault limit. This limit must be high enough to eliminate nuisance following error faults in frequent speed change applications.

i00 allows both positive and negative limit: if you select a positive value, an over-limit condition will cause the SMCC to issue a fault condition. This is a latching fault and requires a reset (Cycle power) of the SMCC. If you select a negative value, an over-limit condition will stop the motion but the SMCC communication remains active. A soft reset command "\$", or hard reset command "\$\$" may be used to clear the fault and re-enable the SMCC to normal operation.

- Set i10 1000 (default) for the following error limit in counts. When the following error exceeds the set value, the SMCC turns on the following error indicator.
- Set i01 100 (default) as an upper limit of following error which is regarded as "in-position". When the following error is within this range, the SMCC turns on in-position indicator.

Once in-position has been detected, the SMCC is ready for the next move regardless of subsequent in-position status.

- Set i11 0 (Rapid Hold Mode) or 1 (Controlled Hold Mode).

Rapid Hold Mode: When i11 is set to 0. Upon HOLD (either "H" command, or hardware logic input HOLD/) command, motion is stopped as rapidly as possible and the motor returns to the position where the HOLD command occurred.

Controlled Hold Mode: When i11 is set to 1. Upon the HOLD command, motion starts to decelerate until stop. The deceleration rate is controlled by i08 (Accel/Decel Time).

- Set i32[i52] 128 (default) as a Normal PWM Limit value. This restricts PHA1 and PHA3 output (analog output current command) voltage to half of the maximum amplitude. Adjustment may be necessary for proper overload protection for the motor.
- Set i33[i53] 64 (default) as a Protective PWM Limit value. When overload is detected, this limit is effective instead of i32[i52] until the overload condition is removed. See Appendix A, i33[i53].
- Set i35[i55] (-1 to 67,000,000) as a positive limit counter value. Set it to 1 to disable this function.
- Set i36[i56] to the value in Range (1 to 67,000,000) as a negative limit counter value. Set it to 1 to disable this function.
- Set i37[i57] to the value in Range (+/- 67,000,000) as a position range. Set it to 0 to disable this function.

3.3.5 Parameters For Special Applications

For most applications, you will not need to alter the following i-parameters. Verify and make corrections only if there is a discrepancy.

- Set i02 0 for Normal Data Processing Mode. When this parameter is set to 0, buffer instructions are stored in a buffer for later RUN. It can be set to 1 when you want immediate execution of buffer instructions. This parameter automatically resets to zero upon the reception of "S" (STEP) or "R" (RUN) command.
- Set i03 1 to enable Handshake in Communication. When handshake is enabled, the processor sends a <LF> character after receiving a <CR> character and the command is processed (or stored). By setting this parameter to 0, handshake can be disabled.
- Set i05 to 487 to set the default servo interrupt period.
- Set i07 to 65498 to set the default time scale. Do not attempt to change i05 and i07. See Appendix A for detailed descriptions of i-parameters.
- Set i14 0 when the on-chip A/D converter is not used (to block A/D converter input).
- Set i28[i48] Handwheel Mode control to 3 to disable handwheel input functions.

3.3.6 Operational Diagnostics and System Tuning

Now, the system is ready for a basic move. Follow the example below.

Example 3.3

COMMAND	EXPLANATION
	Make sure that control bus power is still off.
q	To enable SAVE, the SMCC must be in position.
s	Save the updated parameters
\$\$	Reset the system
A0	Select card #0
p	Obtain the current position POS1. Now turn the shaft one revolution clockwise.
p	Obtain the new position POS2. Let $DIST = POS2 - POS1$. If DIST is positive, clockwise is the positive direction. Otherwise, counter-clockwise is the positive direction. DIST should be close to the "cm" value calculated above. If not, check the encoder resolution again.
	Turn off the power. This time, turn on power with the control bus power on. Wait until the system resets. At this stage, the SMCC should hold the shaft. See NOTE 1 below.
A0	Select card #0 (or any other card number).
O20	Try the open-loop output command. The motor should turn in a forward direction as defined above.
k	Kill the servo, motor will coast to a stop. See NOTE 2 below.
O-20	This time, the motor should reverse direction.
Q	Quit open-loop mode.
J	See if the motor jogs, and check jog speed as calculated by i31[i51] and i06. Check the following error.
j	Try jog in the other direction. See NOTE 3.
k	Kill the servo. (No output to the control.) This command can be issued whenever unpredicted situations occur.

NOTE 1.

- (1) The shaft is holding but the stiffness is very low: Needs tuning. Increase i20 value (proportional gain).
- (2) The shaft oscillates back and forth: Too much gain, or needs damping. Increase i21. If this doesn't help, decrease i20.
- (3) Runaway in one direction: Possibly the encoder connection is backward, or control has an offset. Check the encoder connections and control adjustments.

NOTE 2.

- (4) If the motor rotates backwards, both the encoder and motor connections may be reversed. Check wiring.
- (5) If motor does not turn and tries to hold position, check the four limit switch connections. All four limit switch lines must be grounded to rotate properly.

NOTE 3.

- (5) Delayed Reversing Probably too much following error. Needs correct tuning.
- (6) Following Error Fault occurred Needs correct tuning. Also check to see if i00 parameter is too small.

Five parameters are supplied to tune the system as follows.

- Set i20[i40] proportional gain to 10 (to be adjusted to a proper value upon observation of the motion). The SMCC uses proportional gain to calculate and output an error correction signal in the direction proportional to a observed following error. As i20[i40] increases, the system tends to be stiffer. If the value is too big, the system may oscillate. An unstable or oscillating system will cause excessive heating in the motor and may cause damage. Stabilize an oscillating system as quickly as possible by decreasing i20.
- Set i21[i41], the differential gain constant. Set to 20 initially (to be adjusted to a proper value upon observation of the motion). i21[i41] provides the velocity feedback needed to stabilize the servo loop.
- Set i22[i42], the velocity feedforward gain, to 0 initially. Velocity feedforward can be used to reduce the following error during constant speed motion (refer to Appendix A.)
- Set i23[i43], the integral gain. Set to 10 initially (to be adjusted to a proper value upon observation of the motion). A high integral gain helps to reduce the following error, but at the same time, it may introduce oscillation to the system. You may want to keep this gain low for stability reasons.
- Set i30[i50], the acceleration feedforward gain. Set to zero initially. (refer to Appendix A).
- Set i09 to 1 to enable position integration only in stop, or set i09 to 0 to enable position integration all the time.

For normal operation, it is best to use the integrator at rest only, where it monitors position and ensures zero following error by generating a bias in the SMCC's output signal (to counteract the effects of temperature, friction, off-balance loads, etc.). In velocity control applications, the integrator may be enabled at all times and will strive to automatically maintain zero following error, regardless of changing load and inertia conditions. The disadvantage of having the integrator turned on at all times is that if the system will be operated in frequent start and stopping conditions or fast acceleration and deceleration conditions, then the integrator may generate excessive following error during the accel/decel. Since it will not be able to keep up with the rapidly changing velocity, it may cause undesirable overshoot.

The acceleration feedforward gain allows the user to reduce the following error generated during acceleration and deceleration. However, all other servo gains should be adjusted satisfactorily first and then i30[i50] should be increased gradually (by one) until an acceptable following error is achieved.

The "Monitor" Program provides assistance in tuning the servo system. It utilizes the parabolic SMCC's data capturing capability (Chapter 8) to plot both the desired and actual position data, as well as the velocity and following error.

Now you are ready to test the response of the system. Try the following example commands and readjust the parameters.

Example 3.4

COMMAND	EXPLANATION
Q	Quit from the previous kill command. Check the stiffness of the system. See if it oscillates.
J	Did it run smoother than before? Any excessive following error noticed?
j	Jog backward. At this point, re-adjust various tuning parameters to obtain acceptable performance.

When the motor is connected to the load, the dynamic performance may change significantly. Some refinement on following error limits, acceleration/deceleration time and servo constants may be required. Since application and performance criteria may be quite different, you are free to apply your own techniques. In general, the tuning steps described in this section provides the proper guidelines.

The SMCC has special parameters to compensate for system limitations:

- Set i34 [i54] backlash compensation. This parameter allows the SMCC to compensate automatically for backlash between the motor and the load.
- Set i80. The backlash take up parameter specifies the number of encoder counts per background cycle during the jog and handwheel moves, for detail refer to Appendix A, i80 and i34[i54].

- i60[i61] Deadband specifies a +/-position range from the desired position where the SMCC does not close the servo loop, this is useful in system that has a lot of backlash between the load and the motor.

The following steps are optional. Set these parameters if necessary. For further description on i-parameters, please refer to Appendix A.

3.3.7 Use of External Pulse (Handwheel) Input

- Set i16

i16	Description
0	For standard handwheel operating mode
1	For high speed slave mode
2	For pulse synchronization mode time base use Y handwheel
3	Same as mode 2, but use X handwheel

- Set i27 nn (default of 4096).

This parameter determines the ratio of handwheel counts vs. motor encoder counts, a value of 4096 sets a 1:1 ratio. (See Appendix A, i27)

3.3.8 Use of Analog Command Input

The SMCC has provisions to connect a +/-10V analog signal input to control the speed. It has four different modes of operation, set by i14.

- Set i14 (On-chip A/D converter mode)

i14	Description
0	If the analog command input is not used
1	To control the feedrate override
2	To control the jog speed
3	For velocity command servo (no position control)

3.3.9 Position/Velocity Display Setup

- Set i18 to select the display type according to the following table.

i18	Display Type
0	Single Line 8 Character LED (for each axis)
1	Dual Line 24 Character/Line LCD (for each axis)
2	Single Line 40 Character LCD (for each axis)
3	Dual Line 40 Characters/Line LCD (for dual axes)

- Set i38[i58] 1:1.0 (default) for the position display scaling i38[i58] opens the possibility to use custom units to represent position. The general format is dd:mm.n, where n is the number of displayed decimal digits and

$$\text{Displayed Number} = \text{Count} \times \frac{\text{mm}}{\text{dd} \times 10^{**} \text{n}}$$

Example 3.5

Let cm = 2000 (500 line encoder, with x4 counting) A lead screw with a pitch of 5 rev/in is attached to the motor shaft.

If i38 is set to 10000:100.2, then a 1 inch move corresponds to 10,000 counts, and will be displayed 1.00 (inches).

When i38[i58] takes a non-default value, subsequent input data must be entered with the scaled format.

Example 3.6

X30000 ;Move 30000 counts when i38 = 1:1.0 default unit or 3
;inch in the above example.

X3.00 ;Same instruction when i38 = 10000:100.2

- Set i17 to specify the position/velocity update rate. i17 sets the sample period at which encoder counts are measured. The default is 30000 microseconds. Velocity can be derived by "Delta Position" during the sample period specified by i17.

$$i17 = (60,000,000) / \text{cm}$$

Assume cm = 2000, then i17 = 30000 (default value), velocity will be displayed in RPM.

3.3.10 Home Setup

- Set i04 26 (Range 0 255) to desired speed of home operation.

Home feed rate = (i04/256) i06

COMMAND	DESCRIPTION
i04 26	Homing speed of 10% of max speed (i06)
i04 255	Maximum homing speed
i04 0	Home will not take place

- Set i24[i44] 0 for negative home direction.
1 for positive home direction.
- Set i25[i45] nn (+/- 32767) for home offset
- Set i26[i46] to set home interrupt control bits according to the following table.

COMMAND i26 [i46]	TRIGGERING CONDITION
0	(C/ AND HF/) falling edge
1	(C AND HF/) falling edge
2	HF rising edge
3	UF rising edge
4	(C/ AND HF) falling edge
5	(C and HF) falling edge
6	HF falling edge
7	UF falling edge

Example 3.11

Command	Description
i24 1	When the HOME command is received, the SMCC starts to turn in the positive direction until it sees the home flag bit
i26 1	go high (by Limit Switch), and the encoder C channel
i25100	is detected (Rising Edge). At this point, the SMCC moves 100 counts backward which is the new home position

Refer to Appendix E and F for the correct pinouts for the home flag and user flag.

3.3.11 PLC Enable/Disable

- Set i62 to 1 (Enable PLC Programming) or 0 (Disable PLC programming)

3.4 SIMPLE OPERATIONS

Try the following commands. The assumption is made here that the SMCC is up and running with both the X and Y axis.

Example 3.12

COMMAND	DESCRIPTION
A0	You must address the SMCC at least once after power-on or reset before you can ask it to send any information and data back. If there is more than one SMCC in your system, each card must be individually addressed before information can be sent to or returned from the particular SMCC.
p	This is a position request to the SMCC. It will respond with one positional value for one axis or two positional values for two axes.
f	This is a following error request and will also send one or two values back. Please note that if the position request shows a few counts such as "4" on an axis, the corresponding following error will be a "-4"
J	Observe that the X axis begins to move in the positive direction
p	Use "p" or "f" to monitor the motion's progress. This demonstrates that you may request and send data to the SMCC while it is moving.
j	Jog in the negative direction
i06	The SMCC will send the current value of i06, the maximum reference speed.
i06 20000	20000 is the new reference speed. You may enter any other value, enter a bigger number to increase the speed, a smaller number to decrease the speed.
j	Issue another "j" command to stop the jog motion. Repeating a previous jog command will stop the jog motion. Of course, the "q" command will also stop the motion.
JJY	To jog both X and Y together, notice that both X and Y axes will move and that "X" does not have to be typed in but "Y" must be used.
j	Note that the X axis reverses direction.
jY	Note that the Y axis reverses direction.
jjY	Note that X and Y both stop. You could also use "Q" or "q" to stop motion at any time in any direction. To return to zero position (where the SMCC was before the jog was started) use the "Return to Prejog" command (=).
=	This is the "Return to Prejog" command and the SMCC will begin moving until it reaches "0" position where it started.
v	While the SMCC is returning to its prejog position, you can check the velocity of the motor.
i021	This places the SMCC in MDI mode.
i020	This makes the SMCC exit the MDI mode.

Section 4

Direct Commands

Direct Commands are processed upon the reception of a carriage return (<CR> served as a command line terminator). Some commands ("R", "s", "Z" etc.) will be processed only when the motion is stopped and in-position.

For easy reference, the direct commands are divided into several groups according to their functions. As a convention, parentheses () will denote an optional entry; brackets [] will indicate an equivalent command for the Y axis.

4.1 MOTION COMMANDS

Motion command can conduct a motion directly. Commands such as "R", "S", "\$\$", "\$", "=", "q", "h", "J", and "j" can also be issued from the hardware switches.

R Run(Start): Start executing motion buffer instructions continuously without stopping between steps. To activate the R command, the system must be In-position. So, if the system is in "k"(killed) status, you must first issue a "q" (abort), and then "R" (Run).

S Step: Execute current buffer instruction, stop and advance the buffer pointer one block (step) so that another step command executes the next instruction. The step command can also be used to stop continuous block execution (the R command). A step (block) contains one or more instructions, those instructions that conduct actual motion ("X" or "U" etc.) or equivalent instructions ("DWELL", "WX") etc. are considered as a step. Instructions like F1024, or I20100 are not considered steps.

H Feed Hold (Stop): Stop running the program immediately and hold the current position. To resume motion and complete the current block, use the "R" (Run), or "S" (Step) command. There are two modes of Hold operation Rapid Hold mode, and Controlled Hold mode. See i11 for details.

J(X)[JY] Jog Positive: Jog forward at a rate specified by i06 and i31 [i51]. To stop jogging, use another "J" command or issue the "Q" or "q" command. For simultaneous X and Y axes operation, commands such as "JJY", "jJY", or "jjY" can be issued. Consecutive J commands leads to toggling the jogging condition.

j(X)[jY] Jog Negative: Jog backwards at a rate specified by i06 and i31 [i51]. To stop jogging, use another "j" or issue the "Q" or "q" command.

q Abort Current Move: Aborts the current move. Motion resumes from the next block if an "R" or "S" command is received. Also, if the "=" command is received, the SMCC will complete the current move.

= Return to PreJog Position: Return to the position from which the jog or handwheel movement was started. This command is allowed only when the motion is stopped.

h(X/Y) Home: Return to the home position. The homing direction, speed, and home offset are defined by i04, i24 i26. Note that the "h" command causes both the X and Y axis to go to the home position, while "hX" and "hY" command causes only the corresponding axis to go home.

Q Quit: Quit a running program at the end of the current block. Also used to terminate jogging or the "O" (Open loop output) command.

k Kill Servo: This command sets the error output to zero and opens the servo loop. It is used to disable the control and allow free motion of the motor for learning position or other uses. To restore normal servo operation, use the "Q" or "q" command.

O(X)nn [OYnn] Absolute Open-loop Output: The "O" command forces the SMCC to send a torque command with the user specified amplitude to its velocity error output. The nn has a range of +/- 255, where 0 produces zero volts out and 255 produces 10 volts out (as a current control command input). The "O" command automatically disables the servo loop and is very useful for testing and calibration of the control's velocity loop.

Example 4.1

COMMAND	DESCRIPTION
OX25	Will produce 10% of rated current to make the X axis motor run. Note that this is an open-loop move so the velocity will not be controlled.
OOY25	Same as above but will affect the X and Y axes.

The following commands are for special applications only:

%(nn,(mm)) Feed Rate % Override: The range of nn is 0 to 65535, and the default is the value of i07, or 65498. The optional entry mm is a slope increment number added to the current feed rate at each servo cycle until the new value is reached. Its range is (0 65535), unit in 1/1024 seconds. Feed rate change (accel/decel) can be controlled from 1 millisecond to approximately 65 seconds. % command is used to change the velocity of the motor temporarily by changing the reference velocity as

$$\text{New Reference Speed} = \frac{\%nn}{i07} \times \text{Reference Speed}$$

If mm, or nn is not specified, the last specified values, or default (default nn=i07, mm=64) is used. The equivalent buffer commands for % are the "POTnn" and "SI FWmm" commands which can be written separately.

4.2 PROGRAM CONTROL COMMANDS

\$ Soft Reset: Initiates the phase finding operation and preserves all internal data. Usable to correct phasing change and motor type etc. only. Reloads the "last saved" program.

\$\$ Reset: It essentially causes a complete reset to occur. This does not change the baud rate and card address. To initiate a hardware reset, press the INIT button on the front panel.

Z Zero Set: This command sets the current position to zero. It is allowed only when the motion is stopped. The equivalent buffer instructions are X = 0 and Y = 0.

b(nn) Goto Beginning: Go to the beginning of subroutine nn or the main program if nn is not specified ("b" only). The range of nn is 00 to 63.

PC=v: This command allows the user to control the value of the program counter, thus making it possible to execute any program line by first setting the PC value to the desired line number in the buffer. The range of v is 0 to 9999 and must be between 0 and the last line of the subroutines.

I Learn and Store the X and Y position: This command can be used in teach mode programming, as opposed to normal (or calculated) programming. In teach mode, a desired position can be registered by actually moving the end effector (by jogging, by using a handwheel, or manually) to a location, and issuing an "I" command. The net effect is either to insert a new positional instruction or to replace a current instruction with a new positional instruction. The generation of this positional instruction depends on two learning conditions, "learn" and "learn/replace". In the "learn" condition, the SMCC starts with an empty program buffer and registers a series of points; the other condition is to learn through a program already in the buffer and modify existing positions. The SMCC detects its "buffer full bit", if the buffer is not full, the SMCC does a "learn", otherwise, it does "learn/replace" at whatever point in the program buffer it may be.

Example 4.2

COMMAND	DESCRIPTION
Z	Clear the buffer
F1024	Maximum feedrate
HOME	Go to the preset home position
DWE 1024	Wait for one second
J	Manually move the end effector to the desired position
j	Stop there
I	The SMCC will learn the position and appropriate instruction will be written automatically
DWE 1024	Wait for one second
J	Move to the next desired position
J	Stop jogging
I	Learn this point
HOME	Go back home
END	End of program
LIST	Check the program

You must check and verify the positional instruction written by the I command. Sometimes you have to eliminate the Y axis move command if you do not wish to move the Y axis at that time.

z nn Purge the buffer and reserve nn lines for the main program: If nn is not specified, purge the program and reserve all of the memory for the main program's use (No subroutines can be used). nn is 1 to 5630 (32K RAM). The rest of available memory can then be used to store subroutines and PLC programs.

Example 4.3

COMMAND	DESCRIPTION
z	Reserve all available memory for the main program
z111	Reserve the first 111 lines for the main program and leave the remaining memory for subroutines

s Save: Save the buffer program and the i-parameters in permanent memory (EAROM). This command works only when the on-board DIP switch SW5 is set to true (ON), and the SMCC is at the in-position state. The SMCC prompts with an error signal (Bell) if both conditions are not met.

inn vv Initialization parameter set and query: Set inn parameter as specified by vv. If vv is not specified, then current value of the i parameter is displayed. The corresponding buffer command format is Inn vv. See Appendix A. Glossary of initialization/setup commands.

Pnn = v Set the value of a variable parameter: The range of v is between +/- 2 billion ($2^{31}-1$), and nn is 1 to 15 (zero is not allowed). The equivalent buffer instruction is: SET Pnn = v and RESETP, which resets all parameters to zero.

Example 4.4

COMMAND	DESCRIPTION
P01=10000	Variable P01 is defined
X(P01)	Same as X10000

Cmm = AA Defines Cmm as a pointer to address AA: This can be used to create a lookup table. The equivalent buffer instruction is: SET Cmm = AA. Refer to Appendix D for more information on the SMCC's addresses.

Example 4.5

COMMAND	DESCRIPTION
C12=ffb8	Define C12 as a pointer to the parallel port address (FFB8)

Mnn = Cmm.b Defines the M variable as bit b at address Cmm: Range of b is 0 to 7; nn is 0 to 63; mm is 0 to 31.

The equivalent Buffer instruction is: SET Mnn = Cmm.b.

A specific M function bit can be set by SET nn and reset by RESET nn. See chapter 5

The Following Commands are for Special Applications Only

rAA,nn Read Memory Data: Read nn bytes of data starting at address AA (Hex value).

Example 4.6

COMMAND	DESCRIPTION
r2034,5	Will read the version number (5 bytes of data starting at address 2034)
ra3a4,2	Will read the end address of RAM memory

dAA,nn Formatted Read: Read data and display with format:
Address, Data. Similar to Intel hexadecimal format.

Example 4.7

COMMAND	DESCRIPTION
da414,20	Will display the user's program converted to hexadecimal

wAA,nn Write Memory Data: Write a byte of data nn to address AA (Hex value).

Note: 1. nn can be 0 to 255. If nn is not given, the SMCC will report the one byte of data that is at the given address.

2. All addresses are in hexadecimal format, using 0 to 9 and lower case a to f.

4.3 PROGRAM EDITING COMMANDS

LIST n,m – List m lines of the buffer program starting at line n.

LIST n – List line n only.

LIST – List the whole program.

DELETE n,m – Delete m lines of the buffer program starting at line n.

DELETE – This will have no effect. You must define the lines that you want to delete.

INSERT n – Insert additional line(s) of program starting at line n. To exit the insert mode, use (CR).

REPLACE n – Overwrite program line n. To exit the replace mode, use (CR).

Note: The INSERT or REPLACE commands cause the SMCC to output program line numbers when LIST is used, thus indicating that the SMCC is in edit mode. To exit, simply use (CR). The program may be edited even while the SMCC is running the program. This permits on-line modifications to occur, if needed.

znn Clear the Buffer. – The “z” command can be used only when the SMCC is stopped. Without nn, it will clear the program area while leaving the labels (Subroutines) intact. If the “znn” command is used, the entire program and all subroutine labels are cleared, and n lines (0 to 1790) are allocated to the program area, the rest of the available memory can then be used for labels 00 to 63 (Subroutines) as needed.

MAP n **Reallocate n steps or lines for the main program area.** – n is independent of available memory. All labeled subroutines are moved automatically to create the necessary program buffer space.

Example 4.8 – Demonstrate editing commands

COMMAND	DESCRIPTION
A0	Select A0 card
z50	Clear the program buffer and reserve 50 lines for the main program
FOR 3	Start a loop and execute it 3 times
T2048	Specify a move time of 2 seconds
X1000	Command the X axis to move to position 1000
DWE1024	Wait one second. Note: Commands can be abbreviated by their first three characters.
X0	Return the X axis to position 0
DWE1024	Wait one second
NEX	End of loop
END	End of program
REP0	The SMCC does nothing, but places itself into edit mode. REP is the editor's "REPLACE" command. Program line numbers only show up in edit mode.
LIS	This is the "list" program command. You will read the whole program as follows: 0000:FOR3 0001:T2048 0002:X1000 0003:DWE1024 0004:X0 0005:DWE1024 0006:NEX 0007:END
REP01	We are telling SMCC to replace line 1.
T1024	This entry replaces the old line 1 and the SMCC sends the new line 1 as follows: 0001:T1024
LIS	List the program again. You will read as follows: 0000:FOR3 0001:T1024 0002:X1000 0003:DWE1024 0004:X0 0005:DWE1024 0006:NEX 0007:END
DEL01	Delete line 1

Continued on next page

Example 4.8 – Demonstrate editing commands (continued from previous page)

COMMAND	DESCRIPTION
LIS	List program again 0000:FOR 3 0001:X1000 0002:DWE1024 0003:X0 0004:DWE1024 0005:NEX 0006:END
INS00	Insert a new line before line 0
T1024	This becomes line 0 and the SMCC displays it as 0000:T1024
LIS	List program again, 0000:T1024 0001:FOR0 0002:X1000 0003:DWE1024 0004:X0 0005:DWE1024 0006:NEX 0007:END
s	This “s” command will save the program permanently. The program will remain unchanged until the next “s” is issued.

NOTE: Commands such as “X0Y0”, “U1000V1000” (involves both the X and Y axis) while being entered at one source line, internally it occupies two program lines.

4.4 STATUS QUERY COMMANDS

inn – Send stored initialization/setup parameters.

innH – Send the maximum allowable value for the inn parameter.

innL – Send the minimum allowable value for the inn parameter.

innD – Send the factory set default value for the inn parameter.

ix – Send all initialization/setup parameters.

f – Send the following error.

p – Send the current position.

i38 determines the way that a value will be presented by the p command. If i38 is in default value (i38 1:1.0), then the SMCC displays position in terms of counts.

v Send the present velocity. Units are determined by i17.

Velocity is displayed in terms of RPM units if i17 is set to:

$$i17 = (30,000,000) / \text{cm}$$

where "cm" is the count value that corresponds to one revolution.

? Send the status and error codes

The SMCC sends 8 bytes. Each byte has 8 bits, as status indication to the host. The status bytes are transmitted in Hex-ASCII format.

BYTE 1 (FEB5) Bit 0 Home Y Completed Bit 1 Home X Completed Bit 2 Continuous Move Bit 3 Block Request (MDI Mode) Bit 4 Auto Divide Bit 5 Feed Hold (HOLD) Bit 6 Running Bit 7 Timer Active	BYTE 2 (FEB6) Bit 0 Time Rate Mode Bit 1 Single Step Bit 2 Buffer Full Bit 3 Parabolic Mode Bit 4 Limit Switch Bit 5 Not Ready (X) Bit 6 Not Ready (Y) Bit 7 Not Ready To Move
BYTE 3 (FEB7) Bit 0 Jog Positive (X) Bit 1 Jog Negative (X) Bit 2 Limit Position (X) Bit 3 Following Error (X) Bit 4 I²T Limit (X) Bit 5 Not Stopped (X) Bit 6 Not In Position (X) Bit 7 Zero Command Velocity(X)	BYTE 4 (FEB 8) Bit 0 Jog Positive (Y) Bit 1 Jog Negative (Y) Bit 2 Limit Position (Y) Bit 3 Following Error (Y) Bit 4 I²T Limit (Y) Bit 5 Not Stopped (Y) Bit 6 Not in Position (Y) Bit 7 Zero Command Velocity (Y)
BYTE 5 (FEB9) Bit 0 Edit Mode Bit 1 Insert Mode Bit 2 Extended Read Bit 3 Circle Mode Bit 4 Wait X Bit 5 Wait Y Bit 6 Backlash X Bit 7 Backlash Y	BYTE 6 (FEBA) Bit 0 Auto Divide Request Bit 1 Extended Move Flag Bit 2 Extended Move Flag Bit 3 Home Done Flag Bit 4 CCW Circle Bit 5 Circle > 180 Bit 6 Position Frozen (^V) Bit 7 Subr. Buffer Open
BYTE 7 (FEBB) Bit 0 Servo Pointer Bit 1 Servo Pointer Bit 2 Servo Pointer Bit 3 Dwell Request Bit 4 Home in Progress Bit 5 Dwell in Progress Bit 6 Continuous Move Request Bit 7 Memory Request	BYTE 8 (FEBC) Bit 0 Servo Pointer Bit 1 Servo Pointer Bit 2 Servo Pointer Bit 3 Servo Pointer Bit 4 Servo Pointer Bit 5 Stop Request Bit 6 Servo Pointer Bit 7 Servo Pointer

- Bnn** - Send the line number for label nn, or the size of the program buffer if nn is not specified.
- Bx** - Send the line numbers at which all labels, 00 to 63, are located.
- Pnn** - Send the variable parameter values.
- Px** - Send all variable parameter values for P1 to P15.
- Cmm** - Send the defined memory location address for Cmm.
- Cx** - Send all defined memory location addresses for C00 to C63.
- Mnn** - Send the M function definition.
- Mx** - Send the address and bit for all M functions, M00 to M63. The SMCC supports multiple numbers of M-function I/O's to interface with environments like limit switches, status of other equipment etc. Among them, several M-function I/O's are pre-defined as Mnn variables.

SOFTWARE DEFINITION	DESCRIPTION	PIN SPEC	HARDWARE
M01	Machine Output	#1	(MA01)
M02	Machine Output	#2	(MA02)
M03	Machine Output	#3	(MA03)
M04	Machine Output	#4	(MA04)
M05	Machine Output	#5	(MA05)
M11	Machine Input	#1	(MAI1)
M12	Machine Input	#2	(MAI2)
M13	Machine Input	#3	(MAI3)
M14	Machine Input	#4	(MAI4)
M15	Machine Input	#5	(MAI5)
M16	Machine Input	#6	(MAI6)
M17	Machine Input	#7	(MAI7)

Example 4.9 – Usage of M-function I/O

Command	Description
GOTO+3 IF (M13=0)	IF M13 (Machine input #3) is zero, skip the next two instructions and go ahead. Otherwise, execute the next instruction.
RESET 02	Turn M02 (Machine output #2) off
GOTO 05	Goto subroutine 05
SET 01	Turn M01 (Machine output #1) on
DWELL 1024	Wait one second

Example 4.10 – Definition of Cnn and Mnn

Command	Description
	Definition by direct commands
C01 = ff01	Define C01 variable as SMCC memory address FF01
M01 = C01.5	M01 is defined as bit 5 of that address

4.5 CONTROL CHARACTER COMMANDS

NOTES:

1. All <CNTRL> type commands do not require a <CR> (Carriage Return) to be acted on by the SMCC. They are executed immediately upon reception.
2. All <CNTRL> characters affect all cards on-line at once, regardless of whether they are currently addressed or not.

<CNTRL V>

Upon reception of this command, any on-line SMCC will immediately “froozo” the current X and Y positions and store them for later use. The “frozen” position is indicated on the position display unit and will be sent in response to a “p” command. The “p” command also unfreezes the position display. Operation of the SMCC is not affected by the position freeze command. Only the position data is captured at the time of the <CNTRL V> command reception. It is possible to capture the position data simultaneously on all cards by using the <CNTRL V> command. The position data can then be read one card at a time, by using “A0p”, “A1p”, etc.

<CNTRL M> (Carriage Return <CR>)

This is usually sent to the SMCC by the <ENTER> or <RETURN> key on the computer keyboard. It causes the SMCC to act on the non-control characters it has received since the last carriage return (the action is immediate execution or storage in the program buffer, depending on the type of command.) As a control character, it affects all SMCCs on line, not just the one presently addressed. This is useful for sending simultaneous commands or program lines to multiple cards.

<CNTRL H> (Backspace <BS>)

This is usually sent to the SMCC by the <Backspace> key. It erases the most-recently entered character in the SMCC's line buffer. As a control character, this backspace acts on all SMCCs on-line, not just the one currently addressed. In most cases, only the addressed card will have any characters in the line buffer, but if any other cards do, the last character in their line buffer will be erased also. For instance, typing in "A0X1234A1X5678 <BS> <CR>" would leave "X123" in A0's buffer and "X567" in A1's buffer.

4.6 HANDS-ON COMMAND EXAMPLES

The following examples are intended to introduce the basic SMCC commands that are most frequently used in many applications. It is recommended that the motor shaft be disconnected from the load so that the motor runs freely without hitting any limits. Be sure that the motor is tightly attached to a fixed structure, therefore, eliminating the danger from sudden jerks triggered by the rapid change in acceleration.

Turn the power on. The SMCC will hold the motor shaft at a position and set that position to zero. Try to rotate the shaft (manually), a well-tuned system should hold the position and resist any movement.

Example 4.11 Card Selection and Basic Moves

COMMAND	DESCRIPTION
A0	Card address select command. Selects card #0. If only one SMCC card is used, that card must be card #0.
i38	Check if the SMCC responds with 1:1.0. If not, record the response for later use
i38 1:1.0	Example 4.17 has a more detailed description of i38
J	Jog forward. The motor will jog at the predefined speed until commanded to stop. Notice the direction.
J	Typing the J command again will stop the jogging.
J	Again, jog forward.
Q	Quit also stops the motion.
=	This will move the motor to its prejog position
j	A small j commands the motor to jog backwards. Another j will stop the backwards jogging.
p	This command asks for the current position. The SMCC will respond with the present position represented in counter units. The relation to motor speed (RPM) or actual end effector position will be described later.
v	This command asks for the present speed in counter units per second.
q	This aborts the present move. Note that the SMCC controls or holds shaft position.

Example 4.12 Unit of Position

In an SMCC system, the shaft position of the motor is sensed and converted into encoder counts. This encoder output is converted to a counter value inside the SMCC, and used as position information. The minimum resolution that the SMCC can differentiate is governed by the encoder resolution.

COMMAND	DESCRIPTION
k	This command kills the servo action. The motor is not energized. You should be able to turn the shaft freely. Set the motor shaft in a known orientation so that you can refer to it later.
p	Asks for the present position in counter units. Take note of the position. Now turn the shaft one revolution in the forward direction.
p	Again, request the position. Subtract the previous position value from the current position value. Lets call this value "cm". This is the counter value corresponding to one revolution (mechanical).
q	This command forces the motor to hold the position released by the k command.
p	Again, asks for the current position
Z	Sets the current position to zero.
p	The position should be approximately zero.

Example 4.13 Introduction to I-parameters, Positional Units

Commands starting with i and followed by a two digit number are called initialization commands. We use these commands to set up the system configuration parameters. Once set up, they don't have to be changed frequently. These parameters are permanently stored in the SMCC's memory, and are not lost during power down.

One of the i-parameters, i39, specifies the encoder type, direction and position counting method. Type "i39(CR)" to get your current value of i39 and compare it to the following table.

i39 Value	TYPE	DIRECTION
0	Pulse and Direction	CW
1	x1 A/B Quadrature	CW
2	x2 A/B Quadrature	CW
3	X4 A/B Quadrature	CW
4	Pulse and Direction	CCW
5	x1 A/B Quadrature	CCW
6	x2 A/B Quadrature	CCW
7	x4 A/B Quadrature	CCW

i39 Set encoder type, direction and counter mult. Factor

Note that in A/B Quadrature type Encoder, there are three different kinds multiplication factor of counting Encoder signals. Now, previously measured value of "cm" (Count per Revolution, Mechanical) can be analytically calculated by obtaining the encoder's revolution information and using the equation below:

$$\text{cm} = \text{Encoder Lines per Rev.} \times \text{Counter Multiplication}$$

If you have 500 Lines per Revolution Encoder (A/B Type), and i39 is set to 3(or 7), the "cm" will be 2000.

Example 4.14 – Jog Speed Setup

In this example, we will see how the speed of the jog motion is specified by initialization parameters. The jog speed is determined by two commands: i06 Set Reference Speed and i31 Set Jog Speed Ratio

COMMAND	DESCRIPTION
i06	This command asks reference speed of SMCC in counter per second. Record it down for later use. You can convert this speed in RPM by the following equation: $\text{Reference RPM} = \frac{\text{i06}}{\text{cm}} \times 60$ If i06=2000, then reference speed will be 600 RPM.
i06 60000	1800 RPM when cm=2000
i31	This command asks jog speed relative to the reference speed.
i31 128	Change i31 value to 128. By appending numbers after i commands, you can change parameters temporarily. Jog speed in RPM can be obtained by: $\text{Jog RPM} = \text{Reference RPM} \times \frac{\text{i06}}{\text{cm}}$ Calculate Jog RPM for your system with i31=128.
J	Let motor jog and verify RPM, if possible.
q	Abort move
i31 64	i31 is reduced to half (or 1/4 of Ref. speed).
J	Verify if the speed is reduced.
J	Stop jogging.

Example 4.15 – Following Error

In order to control the shaft position as accurately as possible, the SMCC compares the desired position with the current position obtained from the feedback sensor approximately every millisecond. The following error “FE” is defined by

$$\text{FE} = \text{desired position} - \text{current position}$$

Ideally, it is desired that FE be zero at all times (during the move or hold). But due to the limited capacity of the system, there are some non-zero following errors present, especially during the fast acceleration/deceleration. Also we must give some allowance to the system that following errors less than a certain number of counts can be regarded as no error. This is mainly due to a system's mechanical tolerance.

COMMAND	DESCRIPTION
q	Stops motion
f	Send Following Error in count units. Since motion is stopped, Following Error should be close to zero.
j	Jog
f	During the acceleration or deceleration, you may notice some non-zero Following Error
f	Check FE several times
q	Stops motion

Example 4.16 - Acceleration/Deceleration

The motor must be accelerated or decelerated to start or stop its motion, or to change speed during motion. In the SMCC, the acceleration and deceleration rate can be specified by the i08 parameter. i08 specifies time to accelerate and decelerate from the previous speed to the desired speed, in 1/1024 second units (Roughly in milliseconds).

COMMAND	DESCRIPTION
i08	Asks specific acceleration time.
i08 4096	Accel/Decel time in 4 seconds.
j	Let's see the slow change of speed.
j	Stop
i08 256	Accel/Decel time in 0.25 seconds.
j	See if accelerates fast. Of course, if Accel time is too short, motor cannot follow due to its capacity, and may be tripped by Following Error Fault Limit protection.
q	Stop

Example 4.17 – User's unit of Position

So far, in the above examples, the position or distance was referred with counter units.

Note that we set i38 1:1.0 at the beginning. By changing i38[i58], you can use your own choice of units representing position or distance. i38 has a format of dd:mm.p and position unit is changed to

$$\text{Pos. (New Unit)} = \text{Pos. (count)} \times \frac{\text{mm}}{\text{dd} \times 10^p}$$

where p specifies the number of digits after the decimal point. The following example will help you understanding the above equation. It assumes a system with cm = 100, and the shaft is connected to an end effector linked by a 5 pitch ball screw. So a 1 inch move of the end effector corresponds to 500 count.

COMMAND	DESCRIPTION
J	Let it jog for a while.
J	Stop jogging.
i38 1:1.0	
p	552805 Default value in counter unit
i38 500:1.0	dd=500, mm=1, p=0
p	1106 Divided by 500. Inch Unit, but poor resolution!
i38 500:1.3	
p	1.106 Decimal point moved.
i38 500:10.3	
p	11.056
i38 500:1000.3	
p	1105.610 (inches)

If cm = 2000 (500 line encoder with x4 multiplication factor), and an end effector is connected by a 5 pitch screw. One inch movement of the end effector corresponds to 5 revolution of motor shaft, and 10000 counts in counter unit. If i38 is changed to 10000:1000.3, then for 1 inch of move from the zero position, the SMCC will respond with 1.000 upon issuing the p command.

COMMAND	DESCRIPTION
i38 1000:1000.3	Unit in inches
J	Jog
p	Current position in inches
v	Current velocity in inches per sec.
f	Current Following Error in inches per sec.
Q	Quit Jogging

Example 4.18 – Dual Axis Control

When the SMCC controls a DC motor (Brush Type), one SMCC card can handle two axes of motors, called “X” and “Y” axes. If your system is not configured for two axes, skip this example.

COMMAND	DESCRIPTION
i13	it asks about motor type. If responded with 0 you are controlling brush type DC motor. If not, your system is configured for other type of motor. Proceed only if you get i13 = 0.
i15	This command ask if dual axis is enabled. If responds with 0, you have dual axis system. Go ahead if i15=0.
JX	Same as J command but explicitly specifies X axis.
J	Stops X axis Jogging. J and JX are identical command.
JY	Forward Jog on Y axis.
q	Stops motion.
i31	Asks Jog Speed for X axis
i51	Asks Jog Speed for Y axis. For dual axis system, i20 - i39 defines X axis, while i40 - i59 defines for Y axis.

Buffered instructions are another set of commands which are used to build motion programs. Since they have different mnemonics, the SMCC understands if a given command is a direct command or a buffered instruction. Buffered instructions are normally to be stored in a buffer but can be made to be processed immediately, just like direct commands. This is controlled by i02 command.

NOTE: In this example only, Buffer instructions are marked with an * in front of each instruction.

Example 4.19 – Usage of Buffer Instructions

COMMAND	DESCRIPTION
z	This command empties buffer for main program znn command will purge buffer and reserve space for nn program steps.
Z	Reset Current position be Position Zero.
i381:1.0	Default Positional Scale
i02 1	Specifying i02 = 1 makes Buffered instructions processed immediately. Note that this parameter is automatically reset to 0, after “Q”, or “S” ; command is given. As well as “i02 0” command.
* F 1024	Feedrate (Speed) Instruction. $\text{Speed} = i06 \times \frac{\text{nnnn}}{1024} \text{ (Count / Sec)}$ Programmed Moved after this instruction will be executed with this speed. Sped this time, will be same as Reference Speed set by i06.
* X10000	Move to 10,000 counts position (regardless of current position. X specifies absolute position.
* U10000	Move 10,000 counts from current position. U specifies relative distance.
* X0	Move to Zero position.
i02 0	Now Buffer Instructions will be stored. Type above commands once more. They will not be processed, but stored.
* F1024	
* X10000	
* U10000	
* X0	
* END	This instruction necessary to specify end of Program.
LIST	List the entered program for verification. If not correct, enter “z” command and rewrite the program.
R	Run Direct Command. SMCC will process stored program step by step automatically. This has same effect as pressing RUN button from the front panel.
i38 500:10.1	When i38 parameter changes, parameters in Buffer Program are also changed.
LIST	Verify which parameters are changed.
i381:1.0	Change back!

Example 4.20 – Save Command, RESET

The SMCC uses two different kinds of memory for storage. Initialization parameters that have been changed, or buffer programs in the above example are stored in a volatile memory called RAM. They will be lost upon power down. In order to store these changes permanently, they should be stored in a nonvolatile memory called EAROM (or EEPROM). Data in EEPROM can be changed by the s (save) command, so that the data will be preserved during power down. Upon initialization, this data is recovered to RAM to be used.

COMMAND	DESCRIPTION
ix	This command asks to print all the i parameters stored with a single command. In some computers with LCD display, due to the speed of monitor, you may not be able to get correct information. Try individual inn commands to record parameters for later use.
s	Save updated Initialization Parameters and Buffer program. It might take a second to save all programs.
\$\$	System Reset. Almost has the same effect as Hardware RESET.
A0	Remember to issue address command after reset.
LIST	Lists all the programs stored. Note that Line number starts from 0.

NOTE: RESETTING THE SMCC

The following three RESET commands are supplied to reset without interrupting power. You may use them depending on the situation.

1. Hardware RESET Pressing the INIT Button on the front panel will reset the SMCC completely (same as Powerup RESET). If multiple SMCC's are connected in parallel, all the cards will be reset simultaneously.
2. \$ Command (Software Reset) This command resets the motor type and phasing control. All the other information stored in memory will be preserved.
3. \$\$ Command (Software Reset) This command resets everything except it does NOT reload the switch settings (Baud Rate, Address etc.). All memory contents will be lost.

Re-initializing the SMCC

In emergency situations, the SMCC can be reset to its original default condition. Care should be taken when using the following method because all programs and parameters in the SMCC will be lost. This re-initializing of the SMCC is most often used when an errant PLC program, saved in permanent memory, has dominated the communications channel so much that the user cannot disable or erase the program.

Start the re-initializing procedure by holding the STOP / (Abort) line low and bring the INIT / (Reset) line low and release it. The SMCC will enter a special mode with its outputs disabled for the duration of the initialization procedure. It will briefly display the PROM version and date. The SMCC will then display the message:

RUN FOR PRESENT PARAMS
STEP FOR DEFAULT PARAMS

If you wish to keep the I-parameters and the programs that you currently have saved in permanent memory, take the STRT / (RUN) line low and release it. If you wish to set up the card with the factory default parameters and erase all of the programs in memory, take the STEP / (STEP) line low and release it.

If you have selected the default parameters, you will be presented with several more choices, selectable in the same manner as above.

Make sure that your motors cannot be enabled after the initialization if you are taking the default parameters, because the default values of i39 and i59 may cause a runaway condition.

After you have set the I-parameters, you will enter the "switch test" portion of the initialization. The switch test allows you to check the functionality of the hardware inputs to the SMCC. You will be prompted with a line telling you which input is expected. You can advance to the next stage either by taking that input low (the limits are to be taken high) or by taking the STOP / line low, which circumvents that test.

After the switch test is completed, you will be able to test the encoder and handwheel inputs. The least significant sixteen bits of the counter registers for each input will be displayed. The motors will be disabled, so they may be turned by hand at will and the results observed on the display. When you are done with this portion of the test, take the STOP / line low momentarily and the SMCC will return you to normal operation.

At this point, you have changed the parameters and programs in active memory (RAM), but the non-volatile memory (EAROM) has not been changed. If you want to change the EAROM, you must save the contents of RAM into EAROM with the "s" command.

NOTE: Commands or Instructions longer than 3 characters without counting modifiers or parameters can, in general, be abbreviated to first three characters.

For example:

DWE3	for DWELL 3
REP4	for REPLACE 4
DWE1024	for DWELL 1024
STA	for START

Example 4.21 – Control running a motion program

COMMAND	DESCRIPTION
Z	Set current position to zero.
R	RUN the program and before the execution ends type next command
H	Stop and Hold current position. Upon this command motion can be stopped from anywhere.
R	Run from the stopped position. Wait until program execution is finished.
S	Step Command
S	Do next step.
Q	Used to stop program execution at the end of current motion block.
b	Go to Beginning of Program. SMCC uses a pointer internally to memorize current instruction. Changes pointer to the beginning of program. If “bnn”, pointer will indicate Subroutine nn.

NOTES: H (Hold) command has two modes of operation controlled by i11 initialization parameter. Check which mode is set by typing “i11”, and experiment to verify following description.

Rapid Hold Mode: When i11 is set to 0.

Upon HOLD command, Motion is stopped as rapid as possible, and return to the position where HOLD command occurred.

Controlled Hold Mode: When i11 is set to 1.

Upon HOLD command, motion starts to decelerate until stop. Rate is controlled by i08 (Accel/Decel Time).

Section 5

Buffer Instructions

Buffer instructions are stored in the buffer and will be executed later upon the reception of commands such as “S” (step) or “R” (run), or an external switch closure.

As a convention, a command within a pair of () is an optional entry; A command within a pair of [] is an equivalent Y axis command.

5.1 PROGRAM DEFINITION/CONTROL INSTRUCTIONS

START	Start Continuous Motion. A Continuous Motion Block is defined by starting with the START command, and ending with the STOP command. The SMCC will execute the motion blocks consecutively without stopping. The effect of “START” can be ended by using “STOP” or “END” (if at the end of the program) or by issuing an “S” step command.
STOP	End of Continuous Motion Block. See description for the START instruction.
END	End of program. This instruction must be given at the end of the program. Otherwise the “s” (save) command may not work. This also closes all the unclosed loops, or blocks.
Lnn	Labels are used to define subroutines from L00 to L63. A subroutine may contain any number of program steps and may contain loops and other subroutines, nested up to 16 levels deep. Each subroutine must be terminated by a RETURN instruction.
RETURN	Return from subroutine. A subroutine is defined by the Lnn command. If the subroutine is executed through the GOSUB command, the program sequence will return to the next command in the main program, but if it is executed by the GOTO command, the program will END.
SET Pnn = mm	Set variable number nn as mm. Up to 15 variables can be defined (nn from 1 to 15), and the range of mm is from 8388608 to +8388607. Programs can be written with variables (P01 P15) as long as that variable is predefined. This ability is very effective in programming. For example, a program can be tested with low speed and then be converted to high speed in the real application, The whole process involves only reassigning a value to a variable.

Example 5.1

COMMAND	DESCRIPTION
SET P01=512	Define P01 as 512.
FOR 10	Execute the loop 10 times.
F (P01)	Use the defined variable for the feedrate.
X10000	Move the X axis.
X0	Move it back.
NFXT	End of loop.
END	End of program.

The parameters, P01 to P15, may be used in instructions that require simple data such as X (Pnn) (move X axis) or Y (Pnn) (move Y axis) or F(Pnn) (Set Feed Rate), etc. However, Pnn parameters may not be used with GOSUB, GOTO and instructions that have no data (such as STOP, END, .. etc.). Variable arithmetic operations are also possible within the instructions. Examples of acceptable forms are:

$P_{xx} = (+/- P_{nn})$

$P_{xx} = (P_{nn} +/- P_{mm})$

$P_{xx} = (+/- P_{nn} * P_{mm})$

$P_{xx} = (+/- P_{nn}/P_{mm})$

$P_{xx} = (P_{nn} + P_{mm} * P_{kk})$

$P_{xx} = (P_{nn} +/- P_{mm}/P_{kk})$

$P_{xx} = (P_{nn} + P_{mm}*P_{kk}/P_{jj})$

$P_{xx} = (P_{nn} * P_{mm} / P_{kk})$

Valid Usages:

SET P8 =(P06*P11)

SET P7 =(P8+P4)

SET P1 = 5

SET P2 = (P1)

SET P1 = (P7*P10/P11)

NOTE: Constants are not allowed in variable arithmetic operations.

So, if you are trying to do this:

SET P1=(P1+1)

The equivalent commands would be:

SET P2=1

SET P1=(P1+P2)

NOTE: P variables can be defined as whole numbers only. (No fractional numbers) The SMCC rounds fractional P variables down to the nearest whole number. You can get around this by using the following method.

If you are trying to do this:

SET P1=3.1415 ; P1 = Pi

SET P2=21.33 ; P2 = Diameter of 21.33 inches

SET P3=(P1*P2) ; P3 = Circumference

The equivalent commands would be:

SET P1=31415 ; P1 = Pi * 10000

SET P2=2133 ; P2 = Diameter * 100

SET P4=1000000 ; P4 = 10000 * 100

SET P3=(P1*P2/P4) ; P3 = Circumference rounded down to the nearest whole number

In this example, the error due to rounding is .0082 inches. ($3.1415 * 21.33 = 67.0082$) If this is unacceptable, translate the diameter into encoder counts. The circumference would then be in units of encoder counts with an error of less than one count!

The following commands can be incorporated with the P variables i.e., their data can be replaced by a P variable.

FOR	F	T	t	X	Y	POT	Inn
AX	AY	U	V	:d	WX	WY	^d
tt	DLY	X=	Y=	RX=	RY=	R=	
DWELL		SLEW		SET Pn=		DIVIDE	

The following instructions set the variable P_n to the home captured position (HX, HY) or to the ending position of the present move (AX or AY) based on the last home position. Units are in encoder counts.

SET P_n = HX

SET P_n = HY

SET P_n = AX

SET P_n = AY

RESETP Resets all P variables to zero.

FOR nn Start the loop and execute it nn times. The end of the loop must be specified by the NEXT command. If nn = 0, the loop will be executed forever. The range of nn is 0 to 32767. Loops may contain other loops (nested), as long as each loop is terminated properly by a NEXT command. Loops may also contain instructions that execute other subroutines.

Example 5.2

COMMAND	DESCRIPTION
FOR 10	The outer loop will be executed 10 times
DWELL 1024	Pause for 1 Second
FOR 2	The inner loop will be executed 2 times
U100	Move 100 counts
U-100	Move back
NEXT	Corresponds to FOR 2 (Inner loop)
NEXT	Corresponds to FOR 10 (Outer loop)

NEXT – End of FOR Loop. See FOR nn command for detailed usage.

GOTO +nn (IF (NOT) Condition) – Go forward nn program lines.

GOTO -nn (IF (NOT) Condition) – Go backward nn program lines.

NOTE: GOTO+00 is a useless nonfatal statement. GOTO-00 will lock up the card and should never be used. Do not use the GOTO command in such a manner that an indefinite number of these commands can be executed without an intervening time-based commands (Move or Dwell). The SMCC will lockup because it cannot keep track of time.

For example, do not use : GOTO-00 IF (M11=0)
 Instead, use: DWELL 20
 GOTO-01 IF (M11=0)

The range of nn is 0 to 63. "Condition" will be further explained in the following paragraphs.

GOTO nn IF (Condition) – Go to label nn without saving the return address. The program will terminate when a RETURN is encountered. These commands are executed conditionally. If the logic specified by the IF condition is false, the GOTO command is not executed, and the program executes the next command.

NOTE: If you receive a syntax error when you try to use the command GOTO nn IF(Pxx=1), use the following equivalent commands.

COMMAND	DESCRIPTION
C03=a240	Define C03 as a pointer to address a240 (P01) Any C-variable can be used. The addresses for the P-variables are as follows: P01=a240 P02=a244 P03=a248 P04=a24c P05=a250 P06=a254 P07=a258 P08=a25c P09=a260 P10=a264 P11=a268 P12=a26c P13=a270 P14=a274 P15=a278
GOTO05IF03=1	Equivalent to GOTO05 IF(P01=1). Note that 03 refers to the data that is pointed to by C03. In this case, C03 points to P01.

GOSUB nn IF (Condition) – Call subroutine nn (defined by label Lnn) and save the return address. The program will return to the saved address when a RETURN is executed.

NOTE: If you receive a syntax error when you try to use the command GOSUB nn IF(Pxx=1), use the method explained in the preceding description of the GOTO command. The method is the same, just replace GOTO with GOSUB.

NOTE: The "Condition" format in the above instructions must follow one of the following two forms.

- (1) **IF(NOT) nn CC xxx** – Memory Contents with Immediate Data. NOT is optional. nn is the data read from the address defined by Cnn = AA. xxx denotes immediate data in the range 0 to 255. CC is a conditional operator which has the following forms:

"=" (EQUAL)

">" (GREATER THAN)

"*" (BITWISE LOGICAL AND)

"+" (BITWISE LOGICAL OR),

Example 5.3

COMMAND	DESCRIPTION
C01=ff3a	Define C01 as a pointer to address FF3A (Port)
GOTO 05 IF 01 * 16	If the X axis positive limit is set, then jump to subroutine 05.
HOME	Otherwise, execute HOME.

(2) IF(NOT) CC (AND DD) IF(NOT) CC (OR DD)

CC and DD denotes a conditional test that can have one of two forms:

- (Mnn = 0/1), where Mnn is a bit defined by $Mnn = Cnn.b$.
- (Pnn c X/Y) where nn is a parameter number, X or Y indicates current X or Y axis position, and c is a conditional operator which has following forms:
 - ">" (Greater Than)
 - "<" (Less Than)
 - ">=" (Greater than or equal)
 - "<=" (Less than or equal)

Example 5.4

COMMAND	DESCRIPTION
C04 = ff67	Define C04 as Machine Input Address
M11 = C04.0	Define M11 as Machine Input #1 (bit 0)
M12 = C04.1	Define M12 as Machine Input #2 (bit 1)
P11 = 10000	Define Variable P11 as 10000
GOSUB 02 IF (M11=1 AND M12=0)	If M11 is high and M12 is low then execute subroutine 02 defined by L0 and save the return address
GOTO+ 05 IF NOT (P11 > X)	If the present position of the (X axis) is less than 10000, skip 5 instructions

SET nn (IF (NOT) Condition)

RESET nn (IF (NOT) Condition)

These commands Set (Logical TRUE) or Reset (Logical FALSE) Mnn conditionally. The IF or IFNOT is optional. If it is omitted, then the action is always taken. See the description of the GOSUB command for the format of conditions.

SET Cnn = AA – Defines Cnn as a pointer to address location AA. The range of nn is 063 and creates an address lookup table. Once the address is defined, it is referred to by the Mmm instruction or the CASE instruction in the subsequent program.

SET Mnn = Cmm.b – Defines address location Cmm, bit b as an input or output bit defined by Mnn. Ranges are: nn (063); mm (031); b (07). See Appendix D for the default definitions.

Example 5.5

COMMAND	DESCRIPTION
SFT C01 = ff01	Define C01 as SMCC memory address FF01
SET M01 = C01.5	M01 is defined as bit 5 of that address.

CASE mm, xxxpp – The CASE instruction allows flexibility in programming the SMCC from external sources such as thumbwheel switches or simple switch closures. It establishes a table and then controls which step in the table will be executed, depending upon external inputs.

- mm Specifies the address to be examined (defined by the Cmm Command, 063).
- xxx Defines the size of the lookup table depending on pp.
- pp Can be one of four characters and defines the following:
- “B” or “ ”: Interpret the input data as a binary number.
- “D” : Interpret the input data as a Decimal number. xxx is then 0 99.
- “H” : Interpret the input data in the High 4 Bit Nibble only. (Thumbwheel’s 4 line input connected to 4 MSB Bits.) xxx is then 0 15.
- “L” : Interpret the input data in the Low 4 Bit Nibble only. (Thumbwheel’s 4 line inputs connected to 4 LSB Bits.)
- “d” : Same as “D” with complemented data
- “h” : Same as “H” with complemented data
- “l” : Same as “L” with complemented data
- “b” : Same as “B” with complemented data

NOTE: Complemented data means that when the available data is read at the specified address, it is first complemented (inverted) before being used to jump to the specified line following the Case Command. This allows a simple adoption for thumbwheels without changing the basic Case Command. In this case, thumbwheels is attached to a parallel input port which provides either “true” or “false” outputs (“normally open” or “normally closed”). For detail example, refer to Chapter 7.

When executing the CASE instruction:

The SMCC reads the data at the address specified by Cmm and depending on the data value read, jumps to the specified location on the table, executes that instruction and then executes the instruction which follows at the end of the table. Table entries (instructions) can be as many as 255 steps and are written immediately after the CASE. If the data value read is larger than the number of entries, the table is bypassed and the next instruction is executed.

The instructions in the table can be any instructions, including: Moves, Subroutine GOTOs and jumps, I (parameters), F and T (Feedrate and Time), SET M and SET P, etc. As long as sufficient I/O is available, it is possible to read thumbwheels to selectively change move distances, velocities, servo parameters, program selections, etc.

Example 5.6 – CASE Instruction

Assume a BCD thumbwheel switch is connected to the parallel port at address ffb8.

COMMAND	DESCRIPTION
C20 = ffb8	Define C20 as the parallel port
CASE 20, 10L	Read the Low Nibble (L: Lower 4 bits) of C20 and jump to the following 1 instructions depending on the read value
GOSUB 00	Execute Subroutine 00 and exit this table
GOSUB 01	Execute Subroutine 01 and exit this table
GOSUB 02	Execute Subroutine 02 and exit this table
GOSUB 03	Execute Subroutine 03 and exit this table
GOSUB 04	Execute Subroutine 04 and exit this table
GOSUB 05	Execute Subroutine 05 and exit this table
GOSUB 06	Execute Subroutine 06 and exit this table
GOSUB 07	Execute Subroutine 07 and exit this table
GOSUB 08	Execute Subroutine 08 and exit this table
GOSUB 09	Execute Subroutine 09 and exit this table
*** END OF TABLE ***	
After the execution of the subroutine, the program jumps to here.	
SET M01	Turn ON M-function Output #1

The following instructions are for Special Applications only.

mAA,d	Write Memory Data
nAA,d	Logical AND Memory Data
uAA,d	Logical OR Memory Data
xAA,d	Logical XOR (Exclusive OR) Memory Data
mmAA,d	This is the same as the mAA,d instruction except that it executes immediately and does not require an instruction that sends data to the servo

5.2 MOTION DEFINITION INSTRUCTIONS

Inn - Set Initialization/Setup parameters. Same as Initialization/Setup commands except that the Inn commands are stored in the buffer and thus permit the changing of parameters during program execution.

X = nn [Y = nn] - Set the current position to the value of nn. The unscaled range (i38 set to 1:1.0) is 67108864 to 67108863

Fnn - Set the feed rate.

The range is from 0 to 8 million ($2^{23}-1$).

The actual feedrate is also dependent on i06 value.

Note that Fnn x i06 sets the machine feedrate, and the product should be less than 10^{**9} .

Tnn - Set move time. This command defines the move time of the X and Y axis (from start to the beginning of deceleration) in binary milliseconds. i.e. each unit is equal to 1/1024 seconds. The range of nn is 1 to 8 million ($2^{23}-1$). The move is effective until a new F or T instruction is executed. The total time for the move is nn milliseconds plus the decel time(i08).

tnn - Set the acceleration/deceleration time in nn/1024 seconds. Equivalent to i08 (Set Accel/Decel Time) initialization command. It supersedes i08 for programmed moves, but does not change i08, and does not affect jog moves.

TORQUE - This puts the SMCC in torque-limited mode.

If it is followed by a HOME command, the home cycle changes to a torque-limited home. The home move continues until the following error on each axis exceeds the value in i10. The normal offset move is then made and the new position is set to zero. This can be used to do a homing move against the hard stops of your system (no home input). If the TORQUE command is followed by a move until command (any move followed by ^d), then the offset move is made after the following error on either axis is greater than i10.

If the Torque Command is followed by any other move, the commanded position for that move will not be relaxed if the following error is greater than i10.

LINEAR - Defines linear motion. All subsequent instructions are executed in linear mode until a CIRCLE instruction is executed.

CIRCLE n Defines CW/CCW circular motion.

n = 0 : CW direction with an angle less than 180 degree.

n = 1 : CCW direction with an angle less than 180 degree.

n = 2 : CW direction with an angle greater than 180 degree.

n = 3 : CCW direction with an angle greater than 180 degree.

n = 4 : 3-Point smoother (S-Curve)

Care should be taken not to make a move that is exactly 180 degrees. If a 180 degree move is desired, you should break the move into two 90 degree moves. The SMCC will blend these two moves together to make the resulting 180 degree move.

All instructions following this instruction execute circular or 3-Point smoothed motion until a LINEAR instruction is executed (CIRCLE 0 to CIRCLE 3) or until an F or a T command is executed (CIRCLE 4). The value of n determines the type of circle or 3-Point smoothed motion. The 3-Point smoother is for parabolic units only. Both X and Y radii must be defined before executing any move instructions (can be defined before or after the CIRCLE instruction).

RX = nn – Set the present X radius to nn. The range of nn depends on the scale in effect. The unscaled range is 0 to $(2^{27}-1)$

RY = nn – Set the present Y radius to nn. The range of nn depends on the scale in effect. The unscaled range is 0 to $(2^{27}-1)$.

R = nn – Set the present X and Y (radii) both to nn encoder counts.
No scale is used. The range of nn is 0 to $(2^{23}-1)$.

Example 5.7 – To move in a circular arc from position 0,0

COMMAND	DESCRIPTION
F1024	Define the feedrate
CIRCLE 0	Motion is CW and < 180 degrees.
R=1000	Define the radius
X1000Y1000	Define the final arc point

NOTE: As can be seen from the above example, the arc is completely defined by specifying the present position, target position, radius and direction. To execute circular interpolation, more care and planning is required. The following is an example of circular interpolation being done between two separate cards (each card has two axes). The move starts from 0/0.

Example 5.8 – Circular Interpolation between the X axis on A0 and the Y axis on A1

COMMAND	DESCRIPTION
AAz	Address all cards
F1024	Set the feedrate
R=10000	Radius for X and Y is 10000
CIRCLE 0	Set circle mode
A0 U10000WY0 A1 WX10000V0	Make move in 90 degree steps
A0 U20000WY0 A1 WX0V0	
A0 U10000WY0 A1 WX-10000V0	
A0 U0WY0 A1 WX0V0	
END	End of program

To do interpolation on a single card between the X and Y axis, the program would look as follows:

This program will move in an elliptic arc from position 0/0.

COMMAND	DESCRIPTION
A0z	Address Card A0
F1024	Feedrate to 100%
RX = 1000	Radius of 1000 encoder counts
RY = 2000	Radius of 2000 encoder counts
CIRCLE 0	Clockwise circle
U1000V2000	Make the move in 90 degree steps
U0V4000	Another 90 degrees makes a total of 180
END	End of program

The difference is the introduction of the W (Wait) command which generates a wait time proportional to a corresponding move time, and the fact that even though the Y axis on A0 and X axis on A1 are not used, they must be treated as a “phantom” axis and be told to wait or do nothing. This the reason why the unused axis cannot be used to do a linear interpolation while the other is doing circular or vice versa.

Example 5.9 – Circular Interpolation between the X axis on A0 and the X axis on A1

COMMAND	DESCRIPTION
F1024	Feedrate to 100%
R = 10000	Radius of 1000 encoder counts
A0 CIR0 A1 CIR1	A1's X axis must be reversed
A0 U10000WY0 A1 WX10000V0	Make the move in 90 degree steps
A0 U20000WY0 A1 WX0V0	
END	End of program

Example 5.10 – To move in an elliptic arc using A0, X and Y axes from the present position

COMMAND	DESCRIPTION
F1024	Define the feedrate
CIRCLE 0	Define whether the motion is CCW or CW
RX=1000	Define the X radius
RY=2000	Define the Y radius
X1000Y2000	Define the final arc point

DIVIDE n – Divide the next move into n equal parts, if the length of the move is not divisible by n, the remainder is carried along so that the total move distance will be correct. if the move following the DIVIDE instruction is a circle, then the circle will be divided into n equal parts. However, the motion between the equal parts will be linear and not circular. The range of n is 1 to 65535.

NOTE: If the Run “R” command is used to execute the “DIVIDE” command, the divide function is defeated since motion will occur continuously and will not stop at the division points. Therefore, use the Step “S” command or input when executing the “DIVIDE” command.

The following instructions are for special applications only

POT nn – Feedrate override, this instruction has the same effect as the on-line % command. The range of nn is 0 to 65535

SLEW mm – Feedrate slew control. Equivalent to the direct command , “ %nn,mm”. Range of nn is 0 to 65535, and the default is the value of i07, or 65498. And mm is a slope increment number added to the feed rate during each servo cycle. Its range is (0-65535), and its default is 256, or 1/4 second. The feedrate change can thus be rate controlled from 1 millisecond to approximately 65 seconds.

$$\% = \frac{nn}{i07} \times 100$$

See % command for detailed description.

5.3 MOTION INSTRUCTIONS

HOME – Go to the home position. Upon execution of the HOME command, the shaft moves in the direction defined by i24(Set home direction), with the speed defined by i04 (Home feedrate) until the corresponding switch conditions (set by i26) are met. The SMCC then moves to the offset position defined by i25 (Set home offset). Moves based on the home position can still be made using the AX and AY instructions. The X and Y axes may also be individually commanded to go to the home position by using the “HOME1” (X axis) or “HOME2” (Y axis) commands.

DWELL nn – Wait for nn/1024 seconds (approx. in milliseconds).

Range is 0 to 8 million ($2^{23}-1$).

Xnn [Ynn] – Move the X[Y] axis to position nn relative to position zero, (Absolute Move). The unscaled range is -67108864 to 67108863.

AXnn [AYnn] – Move the X[Y] axis to the position nn units from home. (Absolute Move relative to home.) The unscaled range is -67108864 to 67108863.

Unn [Vnn] – Move the X[Y] axis nn encoder counts from the present position (Relative Move). The unscaled range is -67108864 to 67108863.

WXnn [WYnn] – Wait time (no move) equal to the equivalent time for a move. Active for single or dual axis, linear or circular moves. The wait times are proportional to the F, % and i06 values. The unscaled range is -67108864 to 67108863

NOTE: The WX and WY commands are useful in multi-axis SMCC system programming when an axis on the same or another SMCC card must wait precisely for the duration of another axis' motion. The WX and WY commands will maintain the correct timing even when the feedrate or % commands are used to alter the velocity of the moving axis.

: d – This instruction can be added to the end of any motion instruction (X, Y, AX, AY, U, or V) to define a parabolic move with the ending velocity of d. The value of d has the same meaning as in the “F d” command ($i06 * d/1024$ cts/sec). If d is 1024, then the velocity is i06 counts per second. The range of d is -8388608 to 8388607. If :d is used in a command, F and T are inoperative for that command, and the move is executed in “t” time.

WUT – Wait Until Trigger (Parabolic version 2.78 and newer), The SMCC takes i12 time after it receives a “S” (step) or “R” (run) command before a motion command is sent to the control. This is undesirable in some applications. The “WUT” command is specifically designed to provide a means to overcome this problem and make it possible to start motion nearly instantaneously by transporting the required calculation time to a more convenient time where motion does not have to start.

“WUT” is a buffer command and can be used between buffer instructions, but only when velocity is known to be zero. The reason for this restriction is that “WUT” actually causes the % feedrate to be instantly set to zero.

Example 5.12

COMMAND	DESCRIPTION
	"S" (step) command is issued
X5000	Move starts but ends abruptly because
WUT	WUT is executed at the beginning of decel
X10000	This move is calculated but not executed until a trigger is detected

The above example explains that a "S" command will cause a very abrupt stop during the X5000 move since the end of the move is actually the beginning of the deceleration point, where velocity is still high. The "WUT" should therefore be used only when there is no motion (velocity is zero). For example:

COMMAND	DESCRIPTION
:	"S" (step) command is issued
X5000	Move completes then "S" issued again
U0	This is a dummy move to allow the preceding move to finish
WUT	% is set to zero, X10000 move is calculated, but not executed
X10000	X10000 move can be started by issuing a trigger pulse at the SMCC's Y user flag input. Motion will start within one to two mSec. after the trigger is received.

This sequence would work correctly, and not cause any unwanted jerks since the U0 forces the X5000 move to be completed and then allows the "WUT" to be executed after the velocity is zero. Note that if instead of the "S" (step) commands, an "R" (run) command was used at the start of the sequence, then X5000, U0, and the WUT commands would execute automatically and the X10000 move would be pre-calculated and would be ready for execution using the Y user flag input, or if preferred, by sending a "CNTRL R" command via RS232, the parallel port, or the "bus, since the "CNTRL R" command automatically sets the value of the % command to 65498 (the value of i07) and thus causes the move to occur at full speed.

To summarize the "WUT" command's function:

- 1) It is a buffer command and should be used at a point in the program where velocity is zero to avoid instantaneous velocity change. Its function is to calculate the succeeding move and pass the results to the servo routine thus making it ready to make a move, while at the same time automatically setting the value of "%" to zero and preventing the motion from actually occurring.

-
- 2 To activate the "WUT" command (cause it to pre-calculate the next move), a "S" (step) or "R" (run) command is required, either via communication or control panel input.
 - 3) To cause the activated "WUT" command's move to actually occur, use a hardware input trigger at the Y user flag input (J MACH Pin A10, use i46 to configure the trigger's polarity, etc.) which causes the % value to be set to i07 (default 65498) instantly since the YUFL/input causes an interrupt at the CPU. Motion occurs within one to two mSec. of the trigger's reception.

^d Go Until Interrupt

This instruction can be added to the end of any motion or wait instruction (X, Y, AX, AY, U, V, WX or WY) to define a relative move to be made after an X-axis user-flag interrupt occurs. If the interrupt does not occur before the end of the move, an error 5 is displayed and the motion stops. If a time was specified for the move, then the time will be for the total move including the move after the interrupt. When using timed moves, care must be taken to insure that the interrupt does not occur too close to the end of the move. The minimum time for any offset move will be the acceleration time defined by the t instruction. The scale of d is always encoder counts and has a range of -8388608 to 8388607.

tt n (Parabolic version 2.78 and newer) Triggered Move Calculation Time

In normal operation, the SMCC uses an i12 time of 20 to 50 mSec. or more, depending upon the application. The go until command "^" currently uses i12 time to calculate its new destination after the external trigger is received. The "tt" command is specifically provided for setting the calculation time of the "^" command, and is used as a "buffer" command by the SMCC. Upon power on or reset, the value of i12 is copied into "tt", subsequently the program can set the value of "tt" to that appropriate in the application. For example, "tt25" sets the "^" command calculation time to 25 binary mSec. or (25 x 1000)/1024 mSec. If "tt" is not used, its value will remain that of i12. The minimum allowable value for "tt" is 10 mSec. A value of 20 mSec. is recommended for general purpose use. Please note that if heavy use of background time is made, either because of multiple PLC statements or other reasons, the "tt" time may have to be extended.

CAUTION Erratic motion may result if too small of a value is used for "tt"

q... (Parabolic version 2.79 and newer)

Altered Destination Move In MDI Mode

Sometimes when you command moves for immediate execution, it is desired to break into a move before it is finished and give the card a new destination. This can be done by commanding the new move preceded by a lowercase "q." As soon as the SMCC receives this new command, it will start calculating the altered trajectory to the new destination. It will begin this altered trajectory i12 time after receiving the command.

Example 5.13

COMMAND	DESCRIPTION
i021	MDI mode - execute moves immediately
F100	Set feedrate
X10000	Start move toward X = 10000
qX-5000	If issued while the previous move is still executing, this will cause the SMCC to break into that move and alter the trajectory to proceed to this destination

DLY n – Execute a delay time (no programmed move) of $n/1024$ seconds. This command is essentially equivalent to using the commands “T n” and “U 0”, except that it does not alter the current T value. The delay time starts at the deceleration point of the previous move segment, and the t time for deceleration is included in the delay time, so the next time at zero velocity is $n - t$. If the delay time is shorter than the t time, it will take t time to execute, providing only the deceleration to zero velocity, with no net time at zero velocity.

5.4 HANDS-ON APPLICATION PROGRAMS

Example 5.14 - Usage of Buffer Instructions

COMMAND	DESCRIPTION
z	This command clears the buffer for the main program.
Z	Reset the present position to zero
i381:1.0	Default positional scale
i02 1	Specifying i02 = 1 processes buffered instructions immediately. Note that this parameter is automatically reset to 0, after a “Q”, or “S” command is given as well as a “i02 0” command.
F1024	Feedrate (speed) instruction. Actual speed is: $\text{Speed} = i06 \times \frac{nnnn}{1024} \quad (\text{Count / Sec})$ A programmed move after this instruction will be executed with this speed. The speed will be same as the reference speed set by i06.
X10000	Move to position 10,000 (regardless of the present position). X specifies an absolute position.
U10000	Move 10,000 counts from present position. U specifies a relative position.
X0	Move to position zero
i02 0	Now the buffer instructions will be stored. Type the above commands once more. They will not be processed, but stored for later execution.
F1024	
X10000	

Continued on next page.

Continued from previous page

U10000	
X0	
END	End of the program
LIST	List the entered program for verification
R	Run the program. The SMCC will process the stored program step by step automatically. This has same effect as activating the RUN input.
i38 500:10.0	When the i38 parameter changes, the parameters in the buffer program are also changed.
LIST	Verify which parameters have changed.
i38 1:1.0	Change back!

Example 5.15 Time Specified Move, and Fictitious move

Assuming $cm = 2000$

i06 60000 (1800 RPM)

i08 1024 (1 sec.)

The following two programs will give almost identical motion trajectory:

COMMAND	DESCRIPTION
	** TIME SPECIFIED MOVE **
Z	Start from position zero
T 5120	Move will be finished in 5120/1024 seconds. Note that the specified time is from start to the end of the constant velocity move. So actual time will be 5120/1024 + deceleration time.
X100000	This time, 50000 count distance is moved in 6 seconds. So the speed is half of that in the previous move.
DWELL 1024	Pause 1 sec.
X150000	This time, 5000 count distance is moved in 6 seconds. So speed is half of that in the previous move.
DWELL 2048	Pause 2 sec.
END	
	** FEEDRATE MOVE **
Z	Start from zero
F 341	Now go back to feedrate mode (1/3 of full speed)
t 1024	The t instruction specifies the accel/decel time like the i08 command. The time units are in 1/1024 sec. (roughly milliseconds)
U 100000	Remember, this a relative move. Move 100000 counts backward. This time, accel/decel in 0.5 seconds with full reference speed.
DWE1024	Wait for one second

Continued on next page

Continued from previous page

F170	Reduce speed to half
U 50000	
WX160000	Wait the amount of time equivalent to executing the X160000 command
END	End of program

The T Instruction is very convenient in programming multi-axes synchronization, because the programmer does not have to calculate the speed of each axis.

Example 5.16 – More Instructions, Loop Definition

A loop can be defined by the FOR and NEXT instructions. The program block inside of the loop will be executed as many times as defined in the FOR instruction.

COMMAND	DESCRIPTION
HOME	Go to the home position
F 1024	Set feedrate
FOR 3	Start of a loop. Do this block 3 times. (FOR 0 would put the program in an infinite loop.)
DWELL 1024	Wait 1 second (Time units are the same as in the T command)
AX1000	Absolute move from the home position.
AX0	Go back home.
NEXT	End of the FOR loop
END	End of the program

Example 5.17 Usage of Subroutines

A subroutine is defined by starting with a label command (L nn) and ending with a RETURN instruction. Up to 64 (L00 – L63) subroutines can be defined. A subroutine may contain any number of program steps. It can be nested up to 16 levels deep. Program execution is jumped to a subroutine with GOTO or GOSUB commands.

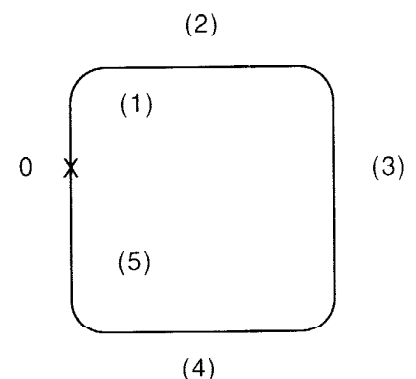
COMMAND	DESCRIPTION
	MAIN PROGRAM
F 512	Set the feedrate
GOTO + 01	Relative jump
U10000	Because of the GOTO + 01 command, this instruction is skipped.
U20000	Move 20000 counts
GOSUB 00	Go to subroutine 00 and remember this line number.
DWELL 1024	Wait 1 second
GOTO 00	Go to subroutine 00 and end the program at RETURN command. Do not remember this line number.
END	End of the main program Subroutine Starts Here
L00F1024	Definition of subroutine 00 and feedrate declaration
X0	Go to position 0
GOSUB 63	Subroutine call from inside of a subroutine
RETURN	Return to the main program or end the program if a GOTO command was given.
L63RET	Dummy subroutine
END	End of subroutines

Example 5.18 Dual Axis Coordinated Motion

The SMCC can control two axes if a brush type DC motor is used. Y axis motion instructions Ynn, Vnn, AYnn (Corresponds to Xnn, Unn, AXnn) can be used in this case. Note that since the execution of the next block is started when the current motion starts decelerating, following example draws a square with round corners.

Example 5.18a

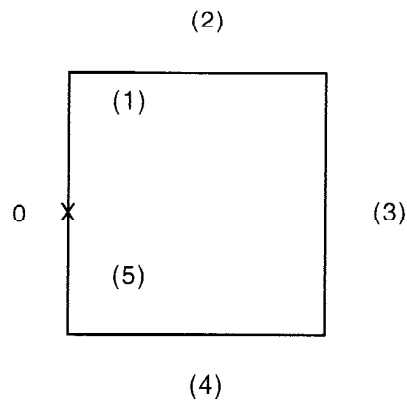
COMMAND	DESCRIPTION
Z	Not a buffer instruction. Sets the present position to zero
F 1024	Set the feedrate to the maximum
Y 25000	Move the Y axis (Path 1)
X 50000	Move the X axis (Path 2)
Y -25000	Move the Y axis back (Path 3)
X 0	Return the X axis (Path 4)
Y 0	Return the Y axis (Path 5)
END	End of program



To draw sharp corners, DWELL 0 (Wait 0 seconds before the next move) instruction may be inserted to allow the next move only after the previous move is finished.

Example 5.18b

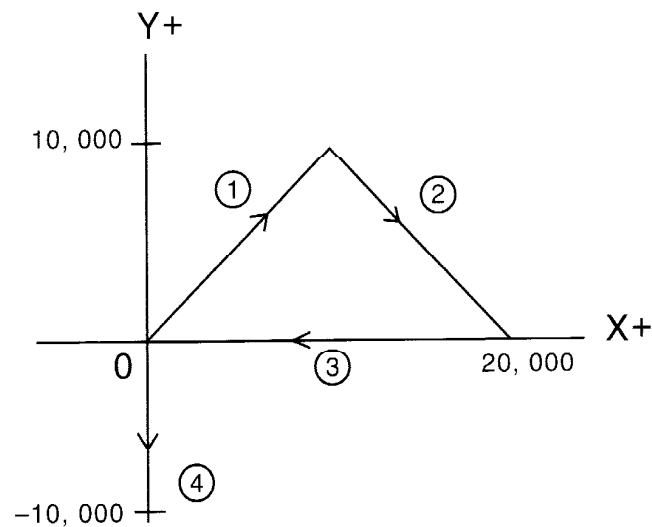
COMMAND	DESCRIPTION
F 1024	Feed rate at maximum
Y 25000	Move the Y axis (Path 1)
DWE 0	Wait until it's done
X 50000	Move the X axis (Path 2)
DWE 0	Wait until it's done
Y -25000	Move the Y axis back (Path 3)
DWE 0	Wait until it's done
X 0	Return the X axis (Path 4)
DWE 0	Wait until it's done
Y 0	Return the Y axis (Path 5)
END	End of program



Example 5.19 - Double Axes Instructions

To execute motion at the same time on both axes, double commands can be used as in this example. In this case, the defined speed is the speed of the diagonal movement. The SMCC will calculate the speed of each axis, so that the two axes of motion is finished at the same time.

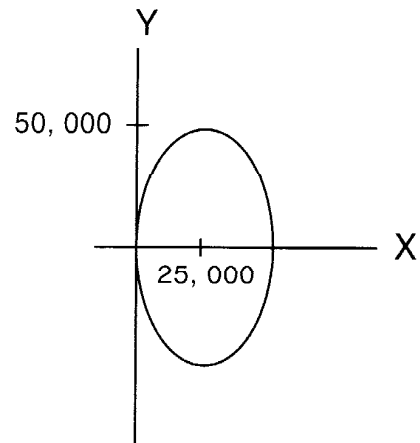
COMMAND	DESCRIPTION
F1024	Full speed
X 10000 Y 10000	X Y axes move simultaneously. (1) 45 degree diagonal motion
U 10000 V-10000	Relative simultaneous motion (2)
DWELL 1024	Wait until the motion is finished
X 0	Move the X axis back (3)
DWELL 0	Wait until the motion is finished
Y 10000	Move the Y axis Back (4)
END	End of program



Example 5.20 – Elliptic Motion.

Elliptic motion can be defined with two principal radii. The following example draws an ellipse from position 0/0.

COMMAND	DESCRIPTION
F100	Define 1/10 speed
CIR 1	Define CCW direction
RX = 25000	Define the X radius
RY = 50000	Define the Y radius
X25000 Y50000	Move to 90 degree point
X0 Y 100000	Keep making 90 degree moves
X-25000 Y50000	Keep making 90 degree moves
X0 Y0	Finish ellipse
FND	End of program



Example 5.21 – Multiple SMCC Coordinated Motion

Suppose that A0 and A1 cards are connected in parallel to the host computer. The card's address can be placed anywhere in the instruction. To draw a diagonal line:

COMMAND	DESCRIPTION
AA	Address all cards
z	Clears buffer for the main program
T 2048	Timed move in two seconds
A0 X 10000 A1 U-5000 AA	Relative simultaneous motion
END	
A0 LIST	Verify the A0 program
A1 LIST	Verify the A1 program
AA R	RUN

Example 5.22 – Variables defined Directly.

Variables can be defined later with direct commands. Consider the following short program.

COMMAND	DESCRIPTION
T10240	Move in 10 seconds
X(P05)	to the position defined by P05
END	End of program

To run the program, you can issue commands such as:

COMMAND	DESCRIPTION
P05	Direct command to ask for the present value
P05=500000	Direct command to define variable
Px	Will list all 15 variables with their defined values.
R	Run command

Many instructions allow the use of variables instead of constants. Also, some arithmetic capabilities using variables are possible, i.e.:

F(P01 + P02)
U(P01 * P02)
AX(P01 / P02)
T(P01 + P02/P03)

Example 5.23 – Machine Input/Output Functions

The SMCC supports multiple numbers of M-function I/O's to interface with environments like limit switches, status of other equipment.. etc. The SMCC's defined M-function I/O hardware is as follows:

M01 Machine Output #1 (MA01)
M02 Machine Output #2 (MA02)
M03 Machine Output #3 (MA03)
M04 Machine Output #4 (MA04)
M05 Machine Output #5 (MA05)
M11 Machine Input #1 (MAI1)
M12 Machine Input #2 (MAI2)
M13 Machine Input #3 (MAI3)
M14 Machine Input #4 (MAI4)
M15 Machine Input #5 (MAI5)
M16 Machine Input #6 (MAI6)
M17 Machine Input #7 (MAI7)

The following instructions are one possible usage for Mnn variables.

COMMAND	DESCRIPTION
GOTO+03 IF(M13=0)	If M13 is FALSE, skip the next three instructions and go ahead. Otherwise, execute the next instruction.
RESET 02	Turn off M02 (M-function Output #2)
DWELL 1024	Wait one second
GOTO 05	Go to subroutine 05
SET 01	Turn on M01 (M-function Output #1)
DWELL 1024	Wait one second
:	
L05	Label for subroutine 05
:	
RETURN	End of subroutine

Example 5.24 – Processor Address Definition (Bits and Bytes)

By using Cnn and Mnn definitions, any address can be defined and used in the program. For example, the corresponding definitions for M01 (M-function output #1, bit5 of address ff01) is:

COMMAND	DESCRIPTION
C01 = ff01	Define C01 variable as SMCC memory address. (Direct command)
M01 = C01.5	M01 is defined as bit 5 of that address. (Direct command)
SET C01 = ff01	Define C01 variable as SMCC memory address. (Buffer command)
SET M01=C01.5	M01 is defined as bit 5 of that address. (Buffer command)

As you can see, you have to know the memory addresses and bit positions of individual functions to be able to define them. See Appendix D for a memory map of the SMCC. Note that since C01 – C05, M01 – M05 and M11 – M17 are already defined internally for the M-functions, it is recommended not to overlay these definitions for different usage.

Section 6

PLC Programming

The PLC (Programmable Logic Controller) instructions are for the purpose of allowing the execution of instructions in conjunction with the SMCC's normal sequence. A PLC program can not perform any motion directly, but it can change variable parameters by logical comparison of position, velocity, following error, or variable parameters (Mnn, Cnn, Pnn). The SMCC can add more power in many applications with the proper utilization of PLC programming.

The advantage of the PLC program is that it has an IF-THEN-ELSE structure and can perform execution on an asynchronous and continuous basis, without affecting the main buffer program. PLC programs are active at all times, even if a motion program is not being executed.

To enable the PLC program, set i62 to 1 (Parabolic) (The default is PLC disabled.)

6.1 DIRECT PLC COMMANDS

PLC – List the PLC program. Similar to the LIST command. It lists all stored PLC program lines on the terminal.

PURGE – Purge the PLC program. This command deletes the PLC program stored in memory. The PLC program cannot be directly edited. To modify a PLC program, "PURGE" it, and reload the modified program. To expedite program editing, invoke the SMCC MONITOR program and use its built-in full-screen editor to modify the PLC program.

6.2 PLC BUFFER INSTRUCTIONS

IF Condition Start of condition, see Note 1

THEN Action Do this if the condition is true, see Note 2

ELSE Action Do this if the condition is not true

AND Condition/Action AND (can be used with a condition or an action)

OR Condition/Action OR (can be used with a condition or an action)

END End of the PLC program

A nested IF-THEN-ELSE can construct an even more intricate PLC program. In general, PLC branches can be constructed by using AND IF, THEN IF, or ELSE IF combinations. An IF statement is automatically translated by the SMCC as a PLC statement and will not be displayed by the LIST command.

Note: **A bad PLC program may trap the SMCC (lock it up). Because of this, it is best to save PLC programs in EAROM (with the "s" command) only after they have been fully debugged. If an error in a PLC program locks up the SMCC, a reset will reload the last saved version of the PLC. If a bad PLC program is saved and then locks up the SMCC, refer to the Re-initializing the SMCC area of section 4.6.**

Every IF condition requires a THEN or THEN IF statement to close the PLC branch. If none of the ELSE statements require an action, the ELSE statement may be omitted and the SMCC will add them automatically. However, if one of the ELSE statements requires an action, then all the previous ELSE statement must be entered in order to properly execute the PLC program (See example 6.3).

AND statements can be used in conjunction with a THEN statement to define multiple actions to be taken when the IF condition is satisfied. They can take any of the actions that a THEN statement can.

An AND statement coupled with an IF statement carried a condition with it, not an action.

The following are valid PLC instructions.

COMMAND	DESCRIPTION
IF C01 = 10	If the data pointed to by C01 = 10
THEN P03 = 1024	Assign a value to the P variable
IF FEX > 2000	If the following error of the X axis > 2000
OR VX < 20000	Or if the velocity of the X axis < 20000
OR P05 = 2048	Or if P05 = 2048
THEN P05 = P05 + 2048	Then ...
AND M06 = 1	And ...
ELSE P05 = 1024	If not true then...
AND I08 = 256	And...
IF M01 = 0	If M01 and M02 are zero, then
AND M02 = 0	
THEN I02 = 1	Set the i02 parameter to a 1
AND SEND "MDI__MODE"	This message is sent to the host and printed on the screen.
AND "i0150"	Note that i parameters may also be changed by sending an on-line command as well as using "I", as seen above.
AND C15,4 = C15 + 15000	Increase the data in the 4 bytes at the address pointed to by C15 by 15000
ELSE.....	
END	End of the PLC program

NOTE: I62 must be set = 1 to enable the PLC program.

Note 1 Condition Formats:

- (a) $M_{nn} = 0/1$ depends on the status of M-function I/O. $nn = 0$ to 63.
- (b) $C_{nn} = vv$ depends on the byte value vv of the SMCC address defined by $C_{nn} =$ (or SET $C_{nn}=$) command. $nn = 0$ to 63, $vv = 0$ to 255.
- (c) $T_n = 0$ True if timer, $n = 0-3$, has counted down to 0.
- (d) $XXXX\ C\ WWW$

Where C can be one of the "<" or ">". C evaluates the value of $XXXX$ and compares with data WWW . $WWW = +/- (2^{31}-1)$ and $XXXX$ takes the format as following:

$C_{nn,m}$ $nn = 0-63$, $m = 1-4$, m bytes of data defined by C_{nn} .

AX/AY X or Y absolute position

VX/VY X or Y velocity value

FEX/FEY X or Y following error

P_{nn} P_{nn} parameter value

Note: In format (d), the "=" (equal to) argument cannot be used immediately following the "IF" statement. In order to check for the equality, use the "AND" evaluation in combination with the less than and greater than arguments.

For example:

IF $P_6 > 4$

AND $P_6 < 6$

THEN ..

These statements check the equality $P_6 = 5$.

NOTE 2 Action Formats:

- | | |
|-----------------------------------|--|
| (a) $M_{nn} = 0/1$ | Set or reset M-function output. |
| (b) $C_{nn,m} = C_{nn,m} +/- WWW$ | Add a constant WWW . |
| (c) $I_{nn} = WWW$ | Modify i parameter. |
| (d) $P_{nn} = C_{pp,q}$ | Assign or modify a variable with address data. |
| (e) $P_{nn} = WWW$ | Assign or modify with constant. |
| (f) $P_{nn} = P_{nn} +/- WWW$ | Increment or decrement P_{nn} by WWW . |
| (g) $C_{nn,m} = WWW$ | Set the memory location C_{nn} , m bytes to a value WWW . $m = 1-4$ |
| (h) SEND "Message" | Any ASCII character except the "space" character may be included in the message, which must be enclosed by quotation ". Message cannot exceed 40 characters. |
| (i) "Command" | Typical commands would be "q" or $I_{nn}VV$... etc. |
| (j) $T_n = SSSS$ | Assign timer n , $n=0-3$, data $SSSS$ has the range of 0-65535. |
| (k) $P_n = AIAN$ (value) | Assign the arctangent of the value to parameter P_n , $n = 0-15$. Value must be one of the following:
(VX/VY) or (VY/VX) or (HX/HY) or (HY/HX) or
(AX/AY) or (AY/AX) or (FEX/FEY) or (FEY/FEX). |

(l) Pn= value

Value must be one of the following:

AX, AY, HX, HY, FEX, FEY, VX, VY.

where: AX, AY: Present absolute position referenced to the most recent power-on, reset, or home position.

HX, HY: Most recently captured position referenced to previous power-on, reset, or home position. FEX, FEY:

Present following error. VX, VY: Present velocity.

In format (h) and (i), Make sure that a condition which causes the command to be issued is immediately set false so that PLC will not send the same message (or perform the same command) consecutively.

A possible use of ATAN is to calculate the velocity vector of the X & Y axis on one SMCC and send the result to another SMCC so it can use the angular value as a command to orient a "Cutting Knife" on the Z axis in the proper direction, always pointing in the velocity vector direction of the X and Y axis.

NOTE 3 Timers Usage:

The SMCC's PLC has four timer ports T0, T1, T2, and T3. T0 and T1 are slow countdown timers, T2 and T3 are fast countdown timers. The clock for T0 and T1 is approximately 23 Hz. or a period of 43.69 milliseconds per count and the clock for the T2 and T3 is approximately 47 kHz or a period of 21.333 microseconds per count.

The format of the timer command is:

Tn= (DATA) where n is a number from 0 to 3 defining the timer to be used and DATA has a range from 0-65535 and is the multiplier for the period of the counter. (DATA) may be replaced by a variable parameter (P01), etc.

To use the timers, it is necessary to set a timer value first by specifying, for example T1 = 500. This sets the T1 timer's time-out value to $500 \times 43.69\text{mS} = 21.845$ seconds. Then you can test for the time-out by using:

IF T1 = 0
THEN.....

The maximum time for T0 and T1 is approximately 47.7 hours. For T2 and T3 it is 1,398 milliseconds. An example of how to use the timers to turn on outputs M01 to M04 is shown on the next page:

COMMAND	DESCRIPTION
IF P1>0	Set P1 = 1 to start cycle
THEN T0=1000	
AND T1=1000	
AND T2=1000	
AND T3=1000	Set all timers to value of 1000 cycles
AND P1=0	Once set, prevent from setting again
IF T0=0	
THEN M01=1	Turn on M01 when T0 has timed out
IF T1=0	
THEN M02=1	Turn on M02 when T1 has timed out
IF T2=0	
THEN M03=1	Turn on M03 when T2 has timed out
IF T3=0	
THEN M04=1	Turn on M04 when T3 has timed out
ELSE	
END	

NOTE: I62 must be set = 1 to enable the PLC program.

The timer's accuracy depends upon the PLC cycle time and can be considered to be 0, + 3 milliseconds max. If it becomes necessary to examine the progress of the timers before time-out has occurred, the following 2 byte memory locations may be read:

T0 ff0a
T1 ff0e
T2 ff88
T3 ff8a

The values in the listed addresses will indicate the progress of the count down cycle.

6.3 PLC PROGRAM EXAMPLES

Example 6.1 – Interface with the main program

SMCC BUFFER PROGRAM	
COMMAND	DESCRIPTION
F 1024	Full feedrate
FOR 100	Do 100 times
GOTO 03 IF(P05 = 1)	If P05 = 1, jump to subroutine
U100	Increment the position 100 counts
NEXT	End of loop
END	End of program
SUBROUTINE	
L03	Label to mark the beginning of subroutine 03
HOME	Body of the subroutine
RETURN	Return to the main program
PLC PROGRAM	
IF M11 = 0	If MAI1 is OFF
AND M12 = 0	And MAI2 is OFF
THEN P05 = 1	Then change P05 to 1
ELSE P05 = 0	Otherwise, P05 = 0
END	End of PLC Program

NOTE: I62 must be set = 1 to enable the PLC program.

Example 6.2 – COUNTER USING M-FUNCTION INPUT (COUNTER):

The counter is triggered on the trailing edge of the M11 input.

COMMAND	DESCRIPTION
A0	Address card A0
PURGE	Clear the PLC program
IF M11=0	Check if M11 switch is toggled. M30 is used as a flag to cause the P1 counter to count once.
THEN IF M30=0	Check if the M30 flag is zero.
THEN M30=1	If true, set M30 flag
AND P1=P1+1	Add 1 to the P1 counter
AND "P1"	Display the value of P1 on the screen
ELSE	
IF M11=1	Check if the M11 switch is released
THEN M30=0	If true, reset the M30 flag
AND IF P1>999	Check if the P1 counter has
AND P1<1001	reached 1000 counts
THEN P1=0	If true, reset the counter
AND SEND "1000- COUNTS"	Send a message to the screen
ELSE	
END	End of PLC program

NOTE: I62 must be set = 1 to enable the PLC program.

Example 6.3 – Use M-FUNCTION INPUTS AS CONTROL PANEL INPUTS:

This PLC program uses the M-function inputs M15, M16 and M17 as “Run”, “Z” and “hX” (home X axis only) commands. To activate any of these inputs, ground the input, otherwise, leave it open.

COMMAND	DESCRIPTION
A0	Address card A0
PURGE	Clear all existing PLC programs
IF M15=1	Check if the M15 switch is toggled
THEN IF P1<1	Check if P1 is zero
AND P1>-1	
THEN”R”	If true, it starts continuous program execution
AND P1=1	Set P1
ELSE	
ELSE P1=0	Reset P1
IF M16=1	Check if the M16 switch is toggled
THEN IF P2<1	Check if P2 is zero
AND P2>-1	
THEN”Z”	If true, set the X and Y position to 0
AND P2=1	Set P2
ELSE	
ELSE P2=0	Reset P2
IF M17=1	Check if the M17 switch is toggled
THEN IF P3<1	Check if P3 is zero
AND P3>-1	
THEN P3=1	If true, set P3
AND”hX”	Starts the X axis on a home cycle
ELSE	
ELSE P3=0	Reset P3
END	End of PLC program

NOTE: I62 must be set = 1 to enable the PLC program.

Example 6.4

This example shows the usage of a PLC program and suggests a way to organize the writing of the program. This example demonstrates the use of the PLC for providing a function which allows a single switch to be used as the end of travel limit switch and as a home switch for precision zero setting. To do so, requires that the switch be wired to both the “Home Flag” and the “Travel Limit” inputs of the SMCC.

This program uses two servo status bits to determine whether to use the switch as a homing flag or a travel limit switch. This will be explained in the program comments.

COMMAND	DESCRIPTION
A0z	Address card & clear the program buffer
SET C07=feb6	Define C07 to be address feb6 (Status Register)
SET C08=ffca	Define C08 to be address ffca (Interrupt flag register)
SET M50=C07.4	Defines M50 to be bit 4 of feb6 (Limit status bit, bit high disables the limit.)
SET M51=C08.6	Defines M51 to be bit 6 of ffca (Interrupt status bit, low if enable)
HOME	Start a home cycle. Choose and set the appropriate parameters for homing, i04 for speed, i24, i25 and i26 X axis, i44, i45 and i46 Y axis. Homing should be in a direction that will drive it towards the limit switch that is wired to both the SMCC's "Home Flag and "Travel Limit" inputs.
STO	Stop until "RUN" or "STEP" command is received again.
FOR0	Start an infinite loop
F1024	Set the feedrate to 100%, this is the beginning of the motion program.
X100000	Starts a back and forth move of 100000 to 0 with a one second dwell at each end.
DWE1024	Wait one second.
X0	Return to zero.
DWE1024	Wait one second.
NEX	End of Loop
END	End of the motion program. Note that a program "end" is used for the program itself. At the very end, another "END" is used for the PLC program.
A0	Address Card
PUR	Purge all the PLC programs stored in the PCL buffer. If the existing PLC program is not purged, the new PLC program will be appended to what is already in the PLC buffer.
i620	Turns off the PLC so that while loading the PLC program, it does not inadvertently create undesirable effects. Remember that the PLC is always active unless turned off, and that the i62 command gets executed on-line and is not stored in the buffer. The SMCC does turn off the PLC while a program is being loaded, but it never hurts to play it safe.
IFM51=0	The IF statement starts the first PLC branch or statement which looks for the condition of M51=0: look to see if there is an interrupt operation in progress, (The home command sets ffca bit 6 low.) If true, then set feb6 bit 4 high (M50=1) thus disabling the limits switches. (The bit automatically resets at the end of each command cycle thus re-enabling the limits.)
THEN M50=1	
ELSE	If a home cycle is not in progress then do nothing. If the ELSE is not typed in, the SMCC will automatically insert it.
END	The END statement finishes the PLC program. If there are other PLC statements, continue with an "IF" which opens a new PLC branch until all PLC statements are completed.

NOTE: i62 must be set = 1 to enable the PLC program.

The following table shows the program alterations needed for the Y axis.

PARABOLIC SMCC		
	X AXIS	Y AXIS
LIMIT STATUS	feb6.4	feb6.4
INTERRUPT STATUS	ffc8.6	ffc8.6

Finally, a counter PLC program is constructed as follows:

COMMAND	DESCRIPTION
	Counter, using M Function input. The counter is triggered on the trailing edge of the MI1 input
A0PURGE	Clear the PLC program
IF M1=0	Check if MI1 is toggled M30 is used as a flag to cause the P1 counter to only count once when MI1 toggles.
THEN IF M30=0	Check if the M30 flag is zero
THEN M30=1	If true, set the M30 flag
AND P1=P1+1	Add one to the P1 counter
AND "P1"	Display the value of P1 on the screen
ELSE	
IF M11=1	Check if MI1 switch is released
THEN M30=0	If true, reset the M30 flag
AND IF P1>999	Check if the P1 counter has reached
AND P1<1001	1000 counts
THEN P1=0	If true, reset the counter
AND SEND "1000-counts"	Send a message to the screen
ELSE	
END	End of the PLC program. Note that the counter may be preset by setting P1 to a starting value.

NOTE: I62 must be set = 1 to enable the PLC program.

Section 7

Application Programs

7.1 Single Axis Torch Cutter

Suppose a SMCC system is installed to operate a torch cutter. The driving system includes a Baldor motor connected to a gear train of 4:1, and a 5 pitch ball screw to convert to linear motion. The motor has a 500 line encoder as a feedback device.

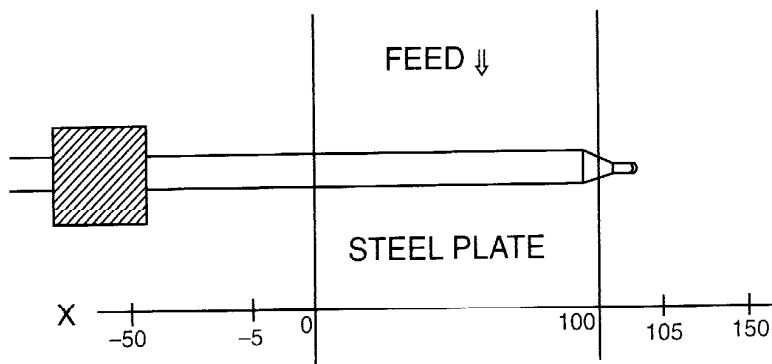
The application system has the following requirements:

- Maximum travel speed (No cut): 180 IPM (Inches per Minute)
- Cutting speed: 45 IPM
- Maximum acceleration: 180 IPM/Sec. = 3 Inches/Sec.²
- Cutting width: 100 inches
- Allowed range of working position +150 inches, -50 inches from position zero
- Handshake:

M11 (MAI1): Feeder ready input (1=Ready, 0=Not Ready)

M01 (MA01): Torch ON/OFF Command (1=Off)

M02 (MA02): Raise/Lower Tip Command (1=Raise)



1. Pre-calculation

- a. One rev. = 2000 counts
- b. One inch move = 20 revs. = 40000 counts
- c. Max. speed 180 IPM = 3600 RPM = 60 RPS = 120000 Counts/Sec.
- d. Cut speed = 45 IPM = 1/4 of max. speed
- e. Accel time from 0 - max. = 1 Sec. (Corresponds to 3 inch move)
- f. Travel during accel to cut speed = 45/60 inch move.

2. Alignment and Setup

Position zero is set. The range of the cutter operation will be -5 to 105, to allow enough travel to accel/decel.

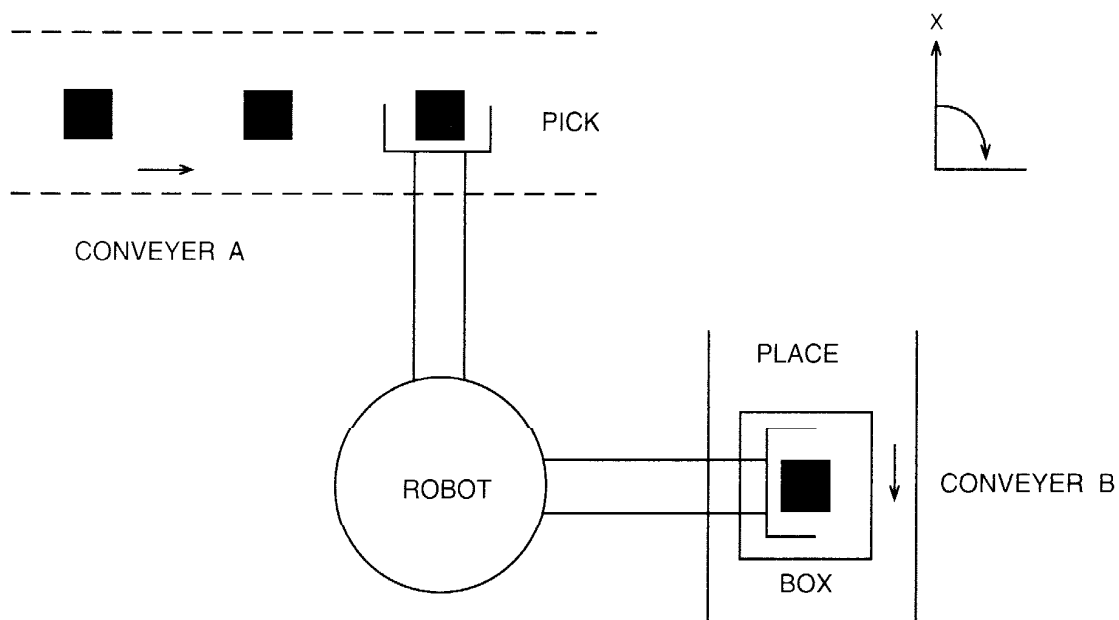
i38:40000:10000.4 Position in inches
Set position zero

3. Program

COMMAND	DESCRIPTION
F1024	Set the feedrate
X-5.0	Starting position
RESET 01	Torch ON
FOR 20	Number of sheets to be cut
SET 02	Tip UP
X105.0	Move to the start position
DWELL 100	Wait for 100 mSec
GOTO-01 IF (M11=0)	Wait until the feeder is ready
RESET 02	Tip down
F256	Cutting speed
X-5.0	Cut
NEXT	End of loop
SET 01	Torch Off
END	End of program

7.2 Two DC Motors in a Pick and Place Operation

Suppose a SMCC system is installed to control a “Pick and Place” robot arm. The robot is configured as a two-dimensional cylindrical coordinate, and has a gripper to grasp and release parts.



Its task is to grasp a part from conveyer A, carry it to the top of the box on another conveyer B, and drop it. The necessary data for each axis are as follows.

X axis: Linear motion
 500 line encoder, 5:1 gear and 5 pitch ball screw

Y axis: Angular motion
 900 line encoder, 20:1 gear reduction

Also, the robot uses the following handshake signals:

M11 (MA11): Part ready to pick up (0 = Ready)
M12 (MA12): Box ready (0 = Ready)
M01 (MA01): Gripper ON/OFF signal (0 = ON)
M02 (MA02): Bell ON/OFF signal (1 = ON)
 Continuous Job done
 Intermittent Help

The programming requirements are as follows:

- X axis units are in inches, Y axis are in degrees.
- Max. speed of motion when unloaded, 1/4 speed with load.
- Rest position is (25 in, 0 deg.)
Move to (100 in, 0 deg.) to pickup parts.
Move to (50 in, 90 deg.) to release parts.

1. System Setup - Assume that the initial setup and tuning is properly done with the following parameters.

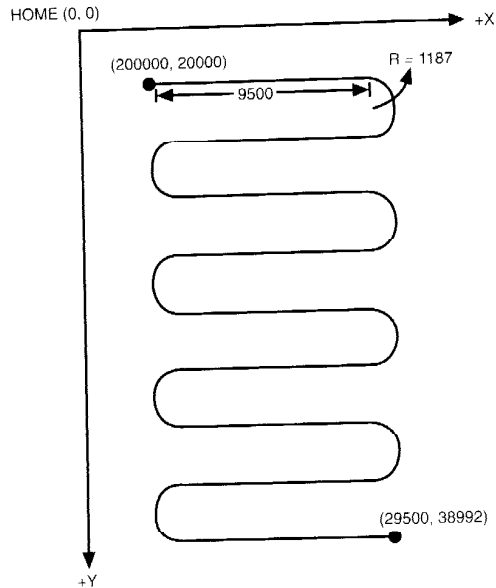
i38 50000:10000:4 (X units in inches)
i58 72000:10000:4 (Y units in degrees)
P01 = 100 Job order issued
Ref. speed and acceleration is set appropriately.

- 2. Program** – After setting the absolute position, and job count command, the operator can RUN ("R") the following program.

COMMAND	DESCRIPTION
SET P03 = 0	Initialize the job count
SET 01	Gripper release
RESET 02	Turn OFF the bell
F1000	Set the feedrate
X25.0Y0.0	Move to the rest position
FOR (P01)	Specified number of times
SET P02 = 0	Reset the time count
F1000	Fast speed
X25.0Y0.0	Move to the rest position
DWELL 100	
GOTO-01 IF(M11 = 1)	Wait for the part ready signal
X100.0	If ready, go
RESET 01	Grasp it
F250	With slow speed
X50.0Y90.0	Move to destination
DWELL 500	
SET P02 = P02 + 1	Increment the time count
GOTO 00 IF(P02>20)	If more than 10 sec. elapsed, jump
GOTO 03 IF(M12 = 1)	See if the box is ready
SET 01	Release the gripper
SET P03 = P03 + 1	Increment the job count
NEX	End of the loop
F1000	Set feedrate
X25.0Y0.0	Go back to the position zero
SET 02	Job done signal
END	End of program
Subroutine 00	
L00	Intermittent bell
FOR 0	Do forever
SET 02	ON for one sec.
DWE1000	
RESET 02	OFF for one sec.
DWE1000	
NEX	End of loop
RET	

7.3 Serpentine Pattern

This program demonstrates the use of subroutines to create a serpentine pattern which uses "line to circle" and "circle to line" connections.



COMMAND	DESCRIPTION
A0z100	Address card A0 and clear the buffer
F324	Set the feedrate
HOM	Go home
t150	Set the accel/decel for 150 mSec.
LIN	Linear mode
X20000Y20000	Move X and Y to position 20000
DWE1024	Wait for one second
SET01	Turn on M01, used as Pen Up/Down control (Pen Down)
DWE100	Wait for about 100 mSec for pen to come down
U9500	Incrementally move X 9500 counts
GOS00	Call Subroutine 00
GOS01	Call Subroutine 01
GOS00	Call Subroutine 00
GOS01	Call Subroutine 01
GOS00	Call Subroutine 00
GOS01	Call Subroutine 01
GOS00	Call Subroutine 00
GOS01	Call Subroutine 01

continued on next page

continued from previous page.

COMMAND	DESCRIPTION
DWE100	Wait for 100 mSec
RES01	Turn off M01 (Pen Up)
DWE100	Wait for about 100 mSec. for pen to go up
X0Y0	Return X and Y to 0 position
END	End of program
SUBROUTINE 00	
L00	Label
CIR1	Circular CCW mode
R=1187	Set the radius to 1187 counts
V2374	Incremental move on diameter
LIN	Linear mode
U-9500	Incremental move X back 9500 counts
RET	Return to the main program
SUBROUTINE 01	
L01	Label
CIR0	Subroutine 01, circular CW mode
R=1187	Set the radius to 1187 counts
V2374	Incremental move one diameter
LIN	Linear mode
U9500	Incremental move X to 9500 counts
RET	Return to the main program

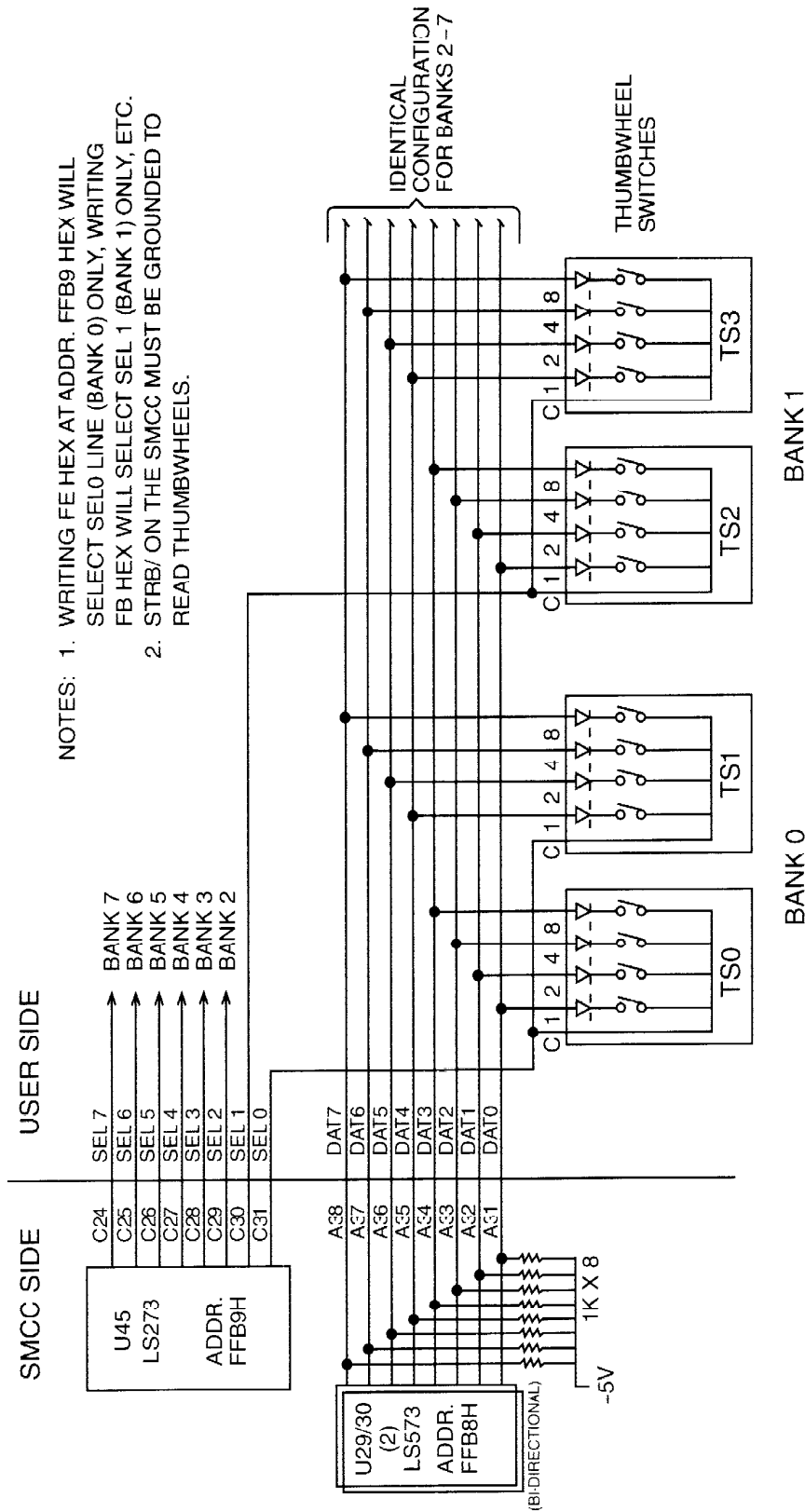
7.4 Thumbwheel Switch Application

The thumbwheel switch outputs are connected to the SMCC's parallel port (Address ff8H). Since only two thumbwheel switch outputs (BCD format) can be read at a time, the connection must have multiplexing capability if more than two digits are desired. SEL0 SEL7 (Address ff9H) connections can be used for multiplexing purposes.

If thumbwheel switches with diodes are connected in series with each BCD output, no additional components are necessary in connecting up to 16 digits of switches. Fig.7.4.1 (page 7-7) illustrates the thumbwheel switch connection without any additional circuitry.

If more than 16 digits are desired, the connection can be made with additional circuitry as illustrated in Fig. 7.4.2 (page 7-8). Depending on the application's needs, the user may design different decoders, or multiplexer circuits. Up to 512 thumbwheel switches can be hooked-up with the necessary decoders and multiplexers.

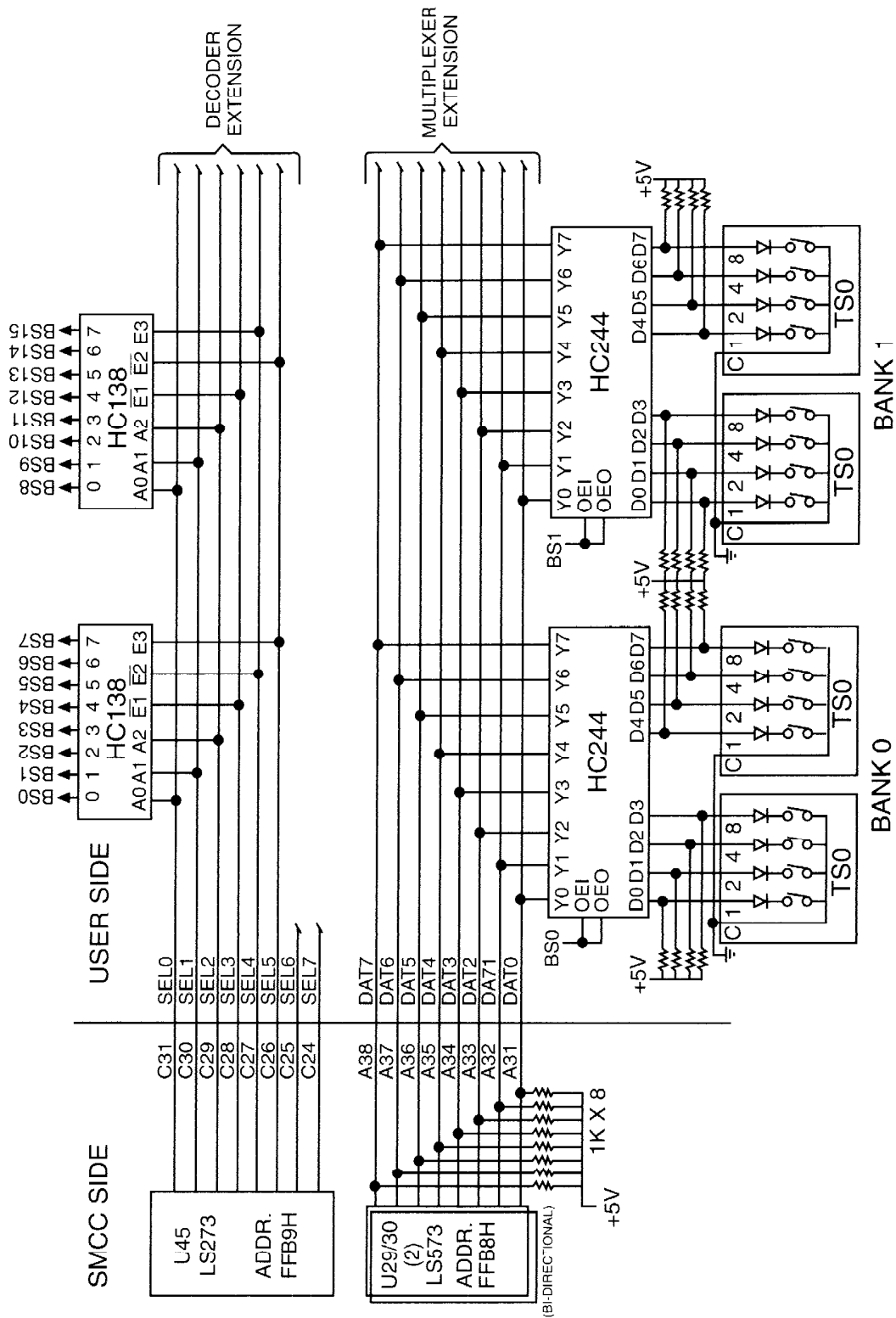
7.4.1 Thumbwheel Switch Connection Without Additional Circuitry



TYPICAL THUMBWHEEL CONNECTION TO SMCC PARALLEL PORT

DIODES, TYP. 64 PLCS. H.P. IN6263 SCHOTTKY (PREFERRED)
ALTERNATE CHOICE, IN4148, (LESS NOISE IMMUNITY NOT
RECOMMENDED.) ALSO, CHANGING PULL-UP SIP RESISTOR
Z21 ON SMCC FROM 1 KOHM TO 4.7 KOHM WILL IMPROVE
NOISE IMMUNITY. Z21 IS SOCKETED.

7.4.2 Thumbwheel Switch Connection With Additional Circuitry



When the thumbwheel switch connection is made, a proper program should be written to sequentially read the switches, process data and convert the input to variables. Examples 7.4.3 and 7.4.4 give you hints on writing thumbwheel switch process routines for your own applications.

Once the thumbwheel switches are connected, and programs written, the operator can set the parameters and press switches to run or stop without using any terminals.

7.4.3 Speed (RPM) Control Using Thumbwheel Switches

Assume the motor encoder has 500 lines and that you use a X4 multiplier, one revolution equals 2000 counts. Attach three thumbwheel switches (TS0, TS1, and TS2) to the circuits given in figure 7.4.2 (use SEL0 and SEL1 for the multiplexing).

Writing the data byte 00 to the SMCC's output port at memory address ffb9 will select TS0 and TS1, writing the data byte 01 will select TS2. Rotates the thumbwheel to get the required RPM.

TS0	TS1	TS2	
3	2	5	set RPM = 325
1	2	0	set RPM = 120

By fetching the data byte at memory location ffb8, you get the data from the currently selected thumbwheel switch. Since the initialization parameter i06 controls the maximum reference speed, as long as you assign the thumbwheel switches' reading to i06 you will have the RPM value ranging from 000 to 999. An internal conversion factor is required to scale the reading from TS0 – TS2. Note the usage of mmAA,d and mAA,d (Refer to Chapter 5), The latter case will be executed only when an instruction sends data to the servo.

2000 (counts/second / 60 = 33 (approx. value)

120 RPM requires to set i06 = 120 x 33

325 RPM requires to set i06 = 325 x 33

COMMAND	DESCRIPTION
A0z	Address card and purge the buffer
SET P06 = 33	Conversion factor
SET P05 = 10	A multiplier
SET P07 = 0	A temporary variable, it will prevent the motor from running at an intermittent value.
F1024	Maximum feedrate
FOR 0	Endless loop
GOSUB 00	Subroutine to read the RPM value
I06(P07)	Set the RPM to the data read from the thumbwheels

continued on next page

continued from previous page

COMMAND	DESCRIPTION
X20000Y20000	Demonstration motions
DWELL 128	
X0Y0	
DWELL 128	
NEXT	
END	
SUBROUTINE 00	
L00	Label
SET C02 = ffb8	Assign the input port for the thumbwheel data
mmffb9, 00	Select TS0 (low nibble) and TS1 (high nibble)
DWELL 50	Set some delay
GOSUB 01	Get the low nibble, P10 contains the nibble
SET P02=(P10 * P05)	Shift one decimal position
GOSUB 02	Get the high nibble, P10 contains the nibble.
SET P02=(P02 + P10)	Now, we get the first and second digit
SET P02=(P02 * P05)	Move one decimal position. TS0 becomes hundreds and TS1 becomes tenths
mmffb9,01	Select TS3 (low nibble)
DWELL 50	Set some delay
GOSUB 01	Get the low nibble, this is the last digit.
SET P02=(P02 + P10)	Add the last digit to the hundreds and tenths
SET P02=(P02 * P06)	Scale to counts/seconds
SET P07=(P02)	
RET	Return to the main program
SUBROUTINE 01	
L01	Label
CASE 02, 10L	Get the low nibble 10 possibilities *
SET P10 = 0	If switch is set to 0
SET P10 = 1	If switch is set to 1
SET P10 = 2	If switch is set to 2
SET P10 = 3	If switch is set to 3
SET P10 = 4	If switch is set to 4
SET P10 = 5	If switch is set to 5
SET P10 = 6	If switch is set to 6

continued on next page

continued from previous page

COMMAND	DESCRIPTION
SET P10 = 7	If switch is set to 7
SET P10 = 8	If switch is set to 8
SET P10 = 9	If switch is set to 9
RET	Return to the main program
SUBROUTINE 02	
L02	Label
CASE 02, 10 H	Get the high nibble 10 possibilities ^
SET P10 = 0	If switch is set to 0
SET P10 = 1	If switch is set to 1
SET P10 = 2	If switch is set to 2
SET P10 = 3	If switch is set to 3
SET P10 = 4	If switch is set to 4
SET P10 = 5	If switch is set to 5
SET P10 = 6	If switch is set to 6
SET P10 = 7	If switch is set to 7
SET P10 = 8	If switch is set to 8
SET P10 = 9	If switch is set to 9
RET	Return to the main program

***NOTE:** L and H are used for BCD “complement” thumbwheels. l and h are used for “non-complement”.

7.4.4 Speed and Job Number Control Using Thumbwheel Switches

Suppose a SMCC system is installed with a 250 line encoder, and a 30 motor revolution per inch linear transducer. One inch of end-effector motion will correspond to 30,000 counts (x4 multiplication factor).

Variables vv, pp, and nn are to be specified by the operator and are as follows:

A. Variable Specifications

- vv: Velocity spec. range of 00 – 60 (Inches per minute) corresponds to 0 to 30,000 Counts/Sec.
- pp: End position spec. range of 0 to 50,000 corresponds to 0 – 1,500,000 counts, indicating 50,000 inches of end-effector position.
- nn: Number of times. Range is from 00 to 99.

R Motion Sequence

Start from position zero, do the following steps nn times

1. Move forward to the position specified by pp with max. velocity (Specified by vv).
2. Wait one second.
3. Move backward to position zero at a speed of 25% of vv.
4. Wait one second
5. Wait until M11 turns ON, and give a pulse to M01.

C. Preliminary Steps

The SMCC Variables are defined as follows, assuming that i38 is set to 1:1:0

P01 = vv (Range: 00 – 60)

P04 = 500 Multiplier to convert vv into counts per sec. To specify max. velocity, I06 (Set reference speed) is calculated by $I06 = P01 * P04$

P02 = pp (Range 00000 – 50000)

P05 = 30 Multiplier to convert pp into counts. The end position will be calculated by $P02 * P05$

P03 = nn (Range 00 – 99)

D. Program with Thumbwheel Switches

Thumbwheel switches are connected as in the following table. It displays a setup of 20 IPM max. velocity, end position of 15.925 inches, and 50 times of task operation.

SPEED		POSITION					NO. JOB	
TS1	TS0	TS2	TS5	TS4	TS7	TS6	TS9	TS8
2	0	1	5	9	2	5	5	0
BANK0		BANK1	BANK2		BANK3		BANK4	

Notes:

1. For convenience, a decimal point can be placed after TS5.
2. TS0 – TS9 are BCD switches (0 – 9, BCD complement logic) with diodes connected in series. For Thumbwheel switch wiring, refer to Fig. 7.4.2.
3. In bank 1, only one switch is connected to DAT0 – DAT3.

The main program calls a subroutine to read and process the thumbwheel switches. Subroutines L10 and L11 read one or two digits from the thumbwheel switches, and this data is processed in subroutine L00.

COMMAND	DESCRIPTION
A0z200	Clear the buffer and allocate 200 program lines in the buffer for the main program
SET P04 = 500	Constants
SET P05 = 30	
GOSUB 00	Read thumbwheel switches
I06 (P01*P04)	Speed spec.
FOR (P03)	Do (P03) times
F1024	Max. velocity
X(P02*P05)	Move to the position specified by P02*P05
DWE1024	Pause for one sec.
F256	25% speed
X0	Return to position zero
DWE1024	Pause for one sec.
GOTO-01 IF (M11=0)	Wait until M11 is ON
SET 01	Give a pulse to M01
RESET 01	
NEXT	End of loop
END	End of the main program
SUBROUTINE TO PROCESS THE THUMBWHEEL SW	
L00	Label
mmffb9,fe	Select bank 0 (Velocity)
GOSUB 11	Read velocity
SET P01 = (P10)	
mmffb9,fd	Select bank 1 (Single digit, MSB)
GOSUB 10	
SET P06 = 100	
SET P02=(P10 * P06)	
mmffb9,fb	Select bank 2
GOSUB 11	
SET P02=(P02 + P10)	
SET P02=(P02 * P06)	
mmffb9,f7	Select bank 3 (LSB)

continued on next page

continued from previous page

COMMAND	DESCRIPTION
GOSUB 11	
SET P02=(P02 + P10)	
mmffb9,ef	Select bank 4 (Number of times)
GOSUB 11	
SET P03 = (P10)	
RET	
SUBROUTINES TO READ THUMBWHEEL SW	
L10	Read one digit from the thumbwheel switch and store it in P10
DWELL 50	Set some delay.
SET C02 = ffb8	Read the parallel port
SET P10 = 0	Reset P10
CASE 02, 10L	Read the low nibble *
SET P10 = 0	If switch is set to 0
SET P10 = 1	If switch is set to 1
SET P10 = 2	If switch is set to 2
SET P10 = 3	If switch is set to 3
SET P10 = 4	If switch is set to 4
SET P10 = 5	If switch is set to 5
SET P10 = 6	If switch is set to 6
SET P10 = 7	If switch is set to 7
SET P10 = 8	If switch is set to 8
SET P10 = 9	If switch is set to 9
RET	Return to the main program
L11	Reads two digit thumbwheel switches and stores it in P10
DWELL 50	Set some delay
SET C02 = ffb8	Read the parallel port
SET P10 = 0	Reset P10
CASE 02, 10L	Read the low nibble *
SET P10 = 0	If switch is set to 0
SET P10 = 1	If switch is set to 1
SET P10 = 2	If switch is set to 2
SFT P10 = 3	If switch is set to 3

continued on next page

continued from previous page

COMMAND	DESCRIPTION
SET P10 = 4	If switch is set to 4
SET P10 = 5	If switch is set to 5
SET P10 = 6	If switch is set to 6
SET P10 = 7	If switch is set to 7
SET P10 = 8	If switch is set to 8
SET P10 = 9	If switch is set to 9
CASE 02, 10H	Read the high nibble
SET P10 = P10 + 0	If switch is set to 0
SET P10 = P10 + 10	If switch is set to 1
SET P10 = P10 + 20	If switch is set to 2
SET P10 = P10 + 30	If switch is set to 3
SET P10 = P10 + 40	If switch is set to 4
SET P10 = P10 + 50	If switch is set to 5
SET P10 = P10 + 60	If switch is set to 6
SET P10 = P10 + 70	If switch is set to 7
SET P10 = P10 + 80	If switch is set to 8
SET P10 = P10 + 90	If switch is set to 9
RET	Return to the main program

***NOTE:** L and H are used for BCD “complement” thumbwheels. l and h are used for “non-complement”.

To run the program, the user must first set variables with the thumbwheel switches, and issue the “R” command, or close the “RUN” switch connection.

7.5 Glue Dispenser Program

This program is used to put glue on in four straight lines with an arc at the end of each line. The part is rotated 90 degrees each time glue is applied to one side.

COMMAND	DESCRIPTION
A0z200	Clear the buffer and allocate 200 program lines in the buffer for the main program
I150	Enable the Y axis
F1024	Set the feedrate to 100% of the i06 value
FOR4	Execute the loop four times
X34000Y10552	Move the glue gun to the starting position
SET01	Turn on M01 output which starts dispensing the glue
DWE256	Wait for 256 mSec.
V750	Move the Y axis incrementally
U320	Move the X axis incrementally
U-2638V21360	Interpolated, simultaneous move
CIR1	Circular mode, CCW (arc setup)
R=2833	Define the radius
U-2925V2565	Arc destination, make the circular move
RESET01	Turn off the glue
LIN	Linear mode, cancel circular mode
U-480	Wipe off the glue tip by moving the X axis
SET02	Turn on the chuck (rotate the chuck 90 degrees)
DWELL50	Wait for 50 mSec.
RESET02	Turn off the chuck (stop it)
X34000Y9732	Move back to the starting position
DWE20	Wait for 20 mSec.
NEXT	End of loop, do this 4 times then
X0Y0	move X and Y back to position 0
END	End of program
A0PURGE	Address card and purge the PLC program in memory
IF M11=0	Check if the emergency stop is on
THEN M01=0	If true, turn off the glue
AND M02=0	Turn off the chuck
AND I03=0	Disable the communications handshake
AND"Q"	Quit the program

continued from previous page

COMMAND	DESCRIPTION
AND I15=1	Disable the Y axis
AND"b"	Set the pointer to the top of the program
AND"h"	Home the X axis
AND P3=1	Set flag 3
AND I03=1	Enable communications handshake
IF M13=0	Check if the X home flag is on
AND P3>0	
THEN I03=0	If true, disable communications handshake
AND"q"	Quit the current move
AND I15=0	Enable the Y axis
AND"h"	Home both the X and Y axes
AND P3=0	Resets flag 3
AND I03=1	Enable communications handshake
IF M14=0	Check if the index chuck is on
THEN M02=1	If true, rotate the chuck
AND T3=2390	Cause the index chuck to stay on for 51 mSec.
AND P5=1	Set flag 5
IF T3=0 and P5>0	Check if T3 is finished with the countdown
THEN M02=0	If true, turn off the index chuck
AND P5=0	Reset flag 5
END	

7.6 Altered Destination On Trigger

This program shows how a SMCC can be programmed to begin a move and upon reception of an external trigger, to alter its destination. The program is treated as a subroutine, which can be invoked by a GOS00 command from the main program.

COMMAND	DESCRIPTION
GOS00	Go to Subroutine 00
...	Remainder of the main program
L00	Subroutine 00 (Label 00)
mffc, 1	Enable interrupts so if a trigger is received by the "Home Flag" or "User Flag" on the X axis, the absolute position of X can be frozen and stored in the HX registers. Note that i20 controls whether the "Home" or "User" flag will generate the interrupt and the polarity of the interrupt. For the Y axis, the control register is ffc8 instead of ffca.
SET C02 = ffca	Define C20 as the address of the capture register.
SET M20 = C20.6	Define M20 as bit 6 of ffca. When this bit is set to zero, the interrupts are disabled. To re-enable the interrupts, use mffc, 1.
T50	Time mode moves, 50 milliseconds per block for all succeeding moves.
RET	End of subroutine, return to the main program.
L01	Subroutine 01 (Label 01)
U1000	Make an incremental move of 1000 encoder counts.
GOTO01 IF(M20=0)	Go to Label 01 if the interrupt has not occurred (trigger has not been received). This conditional statement causes the SMCC to keep returning to Subroutine 01 to make continuous, incremental moves of 1000 counts so long as the trigger has not been received. This motion will go on forever until the trigger is received.

continued on next page

continued from previous page

COMMAND	DESCRIPTION
P1=HX	If the interrupt has occurred, then set parameter P1 to be equal to the captured absolute positional value of X at the time that the trigger occurred. It is important to note that if the interrupt occurs during the approximately 10 to 20 millisecond period preceding each move (servo calculation time), it will not be detected and calculated until the next move. Depending upon the offset distance to travel, this may cause an overshoot and return to final position. The incremental move should therefore be as small as practically possible. The parabolic version actually allows you to specify the calculation time with i12, so that the "Dead Zone" can be precisely predictable.
AX(P1 + P2)	Make a move to absolute position of HX plus an offset move of an amount set by the value of Parameter P2. Note that P2 can be positive or negative.
RET	End of subroutine, return to the main program.

If you wish the trigger search to end at a certain position and not to go on forever, alter L01 and add L02 as follows:

COMMAND	DESCRIPTION
L01	Subroutine 01 (Label 01)
GOTO02IF(X>P5)	Go to Subroutine 02 if the X position is greater than parameter P5.
U1000	Otherwise, make an incremental move of 1000 counts.
GOTO01IF(M20=0)	Loop to subroutine 01 until the interrupt occurs.
P1=HX	
AX=(P1 + P2)	
L02	Subroutine 02
RET	Return to the main program

7.7 Multiple SMCC Sequencing Method

COMMAND	DESCRIPTION
AAq	Sometimes it is necessary to have several SMCCs functioning together in a manner that allows various tasks to be sequenced from one card to the next. The method chosen for passing sequencing information from one SMCC to the next is the use of the SMCC inputs and outputs. On each card, input 1 and output 1 are tied together and then they are tied to all the card I/O 1's. The same applies to I/O 2, 3, & 4. This way any card can output a sequence number from 1 to 16 (4 bits) and any card can read the data. This example is written for 3 cards but up to 16 cards can be handled.
M02=0	Clear all outputs
M03=0	
M04=0	
M05=0	
i1240	Set servo calculation time to 40 mSec.
z100	Clear the buffer
A0GOS01	Start card 0
AAF0R0	Start loop
GOS20	Wait for the start command
A0DWE19	Action for card 0
GOS01	Start card 1
A1DWE20	Action for card 1
GOS02	Start card 2
A2DWE21	Action for card 2
GOS00	Start card 0
AANEXT	End of loop
END	End of the program
L00	Start card A0
GOS16	Wait for all lines to be 1
SET02	Turn on Output 2
SET03	Turn on Output 3
SET04	Turn on Output 4
SET05	Turn on Output 5

continued on next page

continued from previous page

COMMAND	DESCRIPTION
DWE25	Wait 25 mSec
RES02	Turn off Output 2
RES03	Turn off Output 3
RES04	Turn off Output 4
RES05	Turn off Output 5
RET	Return, end of subroutine
L01	Start card A1
GOS16	Wait for all lines to be 1
SET03	Turn on Output 3
SET04	Turn on Output 4
SET05	Turn on Output 5
DWE25	Wait 25 mSec
RES03	Turn off Output 3
RES04	Turn off Output 4
RES05	Turn off Output 5
RET	Return, end of subroutine
L02	Start card A2
GOS16	Wait for all lines to be 1
SET02	Turn on Output 2
SET04	Turn on Output 4
SET05	Turn on Output 5
DWE25	Wait 25 mSec
RES02	Turn off Output 2
RES04	Turn off Output 4
RES05	Turn off Output 5
RET	Return, end of subroutine
L16	
GOT21IF04*15	Wait for all lines to be 1
DWE20	Wait 20 mSec
GOT16	Go back to L16
L20	

continued on next page

continued from previous page

COMMAND	DESCRIPTION
CAS04, 16L	Wait for the start code. The "Case" command is looking at address 4 which is the location of the inputs, see memory map. The input code that makes the A0 card do something is "0", which makes the card execute the L21 routine (a simple return in this case).
A0GOT21	Card A0, If 0, go to L21
DWE20	Card A0, If 1, wait 20 mSec
DWE20	Card A0, If 2, wait 20 mSec
DWE20	Card A0, If 3, wait 20 mSec
DWE20	Card A0, If 4, wait 20 mSec
DWE20	Card A0, If 5, wait 20 mSec
DWE20	Card A0, If 6, wait 20 mSec
DWE20	Card A0, If 7, wait 20 mSec
DWE20	Card A0, If 8, wait 20 mSec
DWE20	Card A0, If 9, wait 20 mSec
DWE20	Card A0, If 10, wait 20 mSec
DWE20	Card A0, If 11, wait 20 mSec
DWE20	Card A0, If 12, wait 20 mSec
DWE20	Card A0, If 13, wait 20 mSec
DWE20	Card A0, If 14, wait 20 mSec
DWE20	Card A0, If 15, wait 20 mSec
A1DWE20	Card A1, If 0, wait 20 mSec
GOT21	Card A1, If 1, go to L21
DWE20	Card A1, If 2, wait 20 mSec
DWE20	Card A1, If 3, wait 20 mSec
DWE20	Card A1, If 4, wait 20 mSec
DWE20	Card A1, If 5, wait 20 mSec
DWE20	Card A1, If 6, wait 20 mSec
DWE20	Card A1, If 7, wait 20 mSec
DWE20	Card A1, If 8, wait 20 mSec
DWE20	Card A1, If 9, wait 20 mSec
DWE20	Card A1, If 10, wait 20 mSec
DWE20	Card A1, If 11, wait 20 mSec

continued on next page

continued from previous page

COMMAND	DESCRIPTION
DWE20	Card A1, If 12, wait 20 mSec
DWE20	Card A1, If 13, wait 20 mSec
DWE20	Card A1, If 14, wait 20 mSec
DWE20	Card A1, If 15, wait 20 mSec
A2DWE20	Card A2, If 0, wait 20 mSec
DWE20	Card A2, If 1, wait 20 mSec
GOT21	Card A2, If 2, go to L21
DWE20	Card A2, If 3, wait 20 mSec
DWE20	Card A2, If 4, wait 20 mSec
DWE20	Card A2, If 5, wait 20 mSec
DWE20	Card A2, If 6, wait 20 mSec
DWE20	Card A2, If 7, wait 20 mSec
DWE20	Card A2, If 8, wait 20 mSec
DWE20	Card A2, If 9, wait 20 mSec
DWE20	Card A2, If 10, wait 20 mSec
DWE20	Card A2, If 11, wait 20 mSec
DWE20	Card A2, If 12, wait 20 mSec
DWE20	Card A2, If 13, wait 20 mSec
DWE20	Card A2, If 14, wait 20 mSec
DWE20	Card A2, If 15, wait 20 mSec
AAGOT20	Address All Cards and Go back to L20
L21RET	Return

7.8 An Automated Encoder Line Counting Program

Count and display the encoder line number

Hardware configuration:

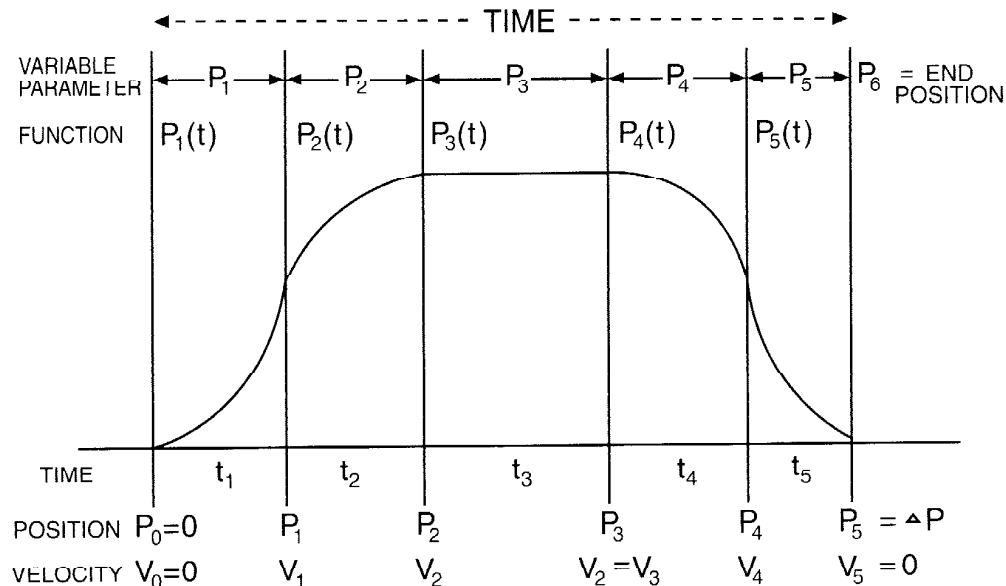
1. Parabolic SMCC.
2. Connect the X-axis encoder C channel output (C/) to the X-axis user flag input, this input pin can be accessed at Baldor's SMCC backplane TB1 pin 14.
3. Servo gains have been properly tuned.

COMMAND	DESCRIPTION
I393	or I397. Multiplied by 4 to ensure stability.
I267	Set the user flag control bit to UF falling
F128	Define the feedrate, avoid low values (such as F32).
U5000^100	Move 5000 counts until interrupt (Triggered by encoder C/). Once the interrupt has been sensed, the motor advances 100 counts. Do not use U5000^0. If no interrupt takes place within 5000 counts, the SMCC will show an error code "5". 5000 is bigger than an ordinary encoder's line count per revolution.
DWE1024	Allows stable servo and following error die out.
SET P2=AX	
U5000^100	We utilize two "U5000^100" commands to make a complete cycle. Do not use the U5000^0 command so that a complete cycle is guaranteed.
DWE1024	Allows stable servo.
SET P3=AX	
SET P3=(P3-P2)	
SET P4=4	
SFT P3=(P3/P4)	
X=0	
U(P3)	LCD's reading equals encoder count.
END	End of the program

Discussion: You can use a PLC program to send P3 (the encoder count) to the computer console or simply type P3 to send the value.

7.9 "S" Curve Accel And Decel (Requires Parabolic Option)

This program allows the user to utilize it as a means of making "S" curve accelerated moves. Six parameters need to be passed to the program in order to make it function properly:



Parameters

- P1 Defines the time to the first velocity inflection point.
- P2 Defines the time from the inflection point to the point where run velocity is reached.
- P3 Defines the run time.
- P4 Defines the time from end of run position to decel inflection.
- P5 Defines the time from decel inflection point to end position.
- P6 Defines the total move distance.

The program is written so as to allow a completely generalized non-symmetrical motion to occur. Parameter definitions can be simplified if the application can use identical accel and decel ramps and a symmetrical inflection point, in other words, $P1 = P2 = P4 = P5$ and by adding the following to the beginning of the program:

```
SET P1 = VALUE
SET P3 = VALUE
SET P6 = VALUE
SET P2 = (P1)
SET P4 = (P1)
SET P5 = (P1)
```

NOTE: i06 must be set to 1024 in order to keep time scaling in lines/sec.

A move can then be made by defining P1 = the inflection point time value in binary milliseconds, P3 = the run time value, and P6 = the total distance value in encoder counts. The minimum time that can be specified for any of the time parameters is 15 milliseconds. The advantage of "S Curve" acceleration is that it minimizes da/dt, rate of change of acceleration, also known as "Jerk" and as a result provides much smoother operation and reduces wear and tear on the machinery.

COMMAND	DESCRIPTION
A0z100	Clear the buffer
i061024	Set the feedrate for lines/sec.
SETC20=febd	Set the address for M20
SETM20=C20.5	Set M20 to save the data for plot
SETP12=(P1+P12)	$P12 = t1+t2$
SETP14=(P4+P5)	$P14 = t4+t5$
SETP7=(P1*P1)	$P7 = t1*t1$
SETP11=(P5*P5)	$P11 = t5*t5$
SETP8 = (P12+P2)	$P8 = t1+2*t2$
SETP10=(P14+P4)	$P10 = 2*t4+t5$
SFTP13=(P8+P10)	$P13 = t1+2*t2+2*t4+t5$
SETP13=(P13+P3)	$P13 = t1+2*t2+t3+2*t4+t5$
SETP13=P13+(P3+P3)	$P13 = t1+2*t2+3*t3+2*t4+t5$
SETP12=(P12*P13)	$P12 = (T1+T2)*T1+2*T2+3*T3+2*T4+T5$
SETP14=(P14*P13)	$P14 = (t4+t5)*(t12+2*t22+3*t3+2*t42+t5)$
SETP7=(P6*P7/P12)	$P7=DP1=DP*t1*t1/(t12+t2)*(t1+2*t2+3*t32+2*t42+t5)$
SETP11=(P6*P11/P14)	$P11 = DP5=DP*t5*t5/(t4+t5)*$ $(t12+2*t22+3*t32+2*t42+t5)$
SETP8 =(P6*P8/P13)	$P8 = (DP12+DDP2)=DP*(t12+D2*t2)/$ $(t12+2*t22+3*t32+2*t42+t5)$
SETP10=(P6*P10/P13)	$P10 = (DP4+DP5)=DP*(2*t42+t5)/$ $(t1+2*t2+3*t3+2*t4+t5)$
SETP9 =(P8+P10)	$P9 = (DP1+DP2) + (DP4+DP5)$
SETP9 =(P6-P9)	$P9 = DP3 = DP - (DP1+DP2) - (DP4+DP5)$
SETP8 =(P8-P7)	$P8 = DP2 = (DP1+DP2) - DP1$
SETP10=(P10-P11)	$P10 = DP4 = (DP4+DP5) - DP5$
SETP15=1024	Time scale in binary mSec.
SETP13=(P15*P9/P3)	$P13 = V2 = V3 = DP3/t3$

continued on next page

continued from previous page

COMMAND	DESCRIPTION
SETP15=3072	3*Time scale in binary mSec.
SETP12=(P15*P7/P1)	$P12 = V1 = 3*DP1/t1$
SETP14=(P15*P11/P5)	$P14 = V4 = 3*DP5/t5$
FOR0	Start the loop
STA	Start run mode
SET20	Start data store
t(P1)	$t1 = P1$
U(P7) : (P12)	$DP = DP1 = P7 : V = V1 = P12$
t(P2)	$t2 = P2$
U(P10): (P13)	$DP = DP2 = P8 : V = V2 = P13$
t(P3)	$t3 = P3$
U(P9) : (P13)	$DP = DP3 = P9 : V = V3 = P13$
t(P4)	$t4 = P4$
U(P10): (P14)	$DP = DP4 = P10: V = V4 = P14$
t(P5)	$t5 = P5$
U(P11): 0	$DP = DP5 = P11: V = 0$
STO	STOP
STA	Start the run mode
SET20	Start the data store
t(P1)	$t1 = P1$
U(-P7) : (-P12)	$DP = -DP1 = -P7 : V = -V1 = -P12$
t(P2)	$t2 = P2$
U(-P8) : (-P13)	$DP = -DP2 = -P8 : V = -V2 = -P13$
t3(P3)	$t3 = P3$
U(-P9) : (-P13)	$DP = -DP3 = -P9 : V = -V3 = -P13$
t(P4)	$t4 = P4$
U(-P10): (-P14)	$DP = -DP4 = -P10: V = -V4 = -P14$
t (P5)	$t5 = P5$
U(-P11):0	$DP = -DP5 = -P11: V = 0$
STO	Stop
END	End of Loop and Program

Section 8

Technical Information

This chapter outlines some important features provided by the parabolic SMCC.

8.1 PARABOLIC PROFILING

The term “parabolic” refers to the parabolic SMCC’s capability to generate parabolic (quadratic, or second-order) velocity profiles, which yield cubic (third-order) position trajectories. Third order in trajectories gives the parabolic SMCC much greater flexibility for the blending of moves, smooth starting and stopping, and close control of the path of movement.

Even when executing lower-order trajectories, the parabolic SMCC generates the trajectories internally using third-order position equations, the low-order trajectories (e.g. trapezoidal moves) just being special cases of the general equations. For instance, a standard point-to-point move of time ‘T’ from point A to point B with acceleration and deceleration times of ‘t’ is broken into three parts: accel, slew, and decel. The acceleration portion lasts for time ‘t’, with constant acceleration ($da/dt=0$) up to the slew speed. The slew portion has constant velocity ($acccl=0$; $da/dt=0$) up until time ‘T’ (duration $T-t$), and the deceleration portion slows down at a constant rate to zero velocity at point B and time ‘T+t’ (duration t). All the key factors in the equations are calculated automatically.

It is important to note that the trajectories described in this section are all commanded trajectories, not necessarily the actual trajectories that the system will follow. It is up to the feedback and feedforward parameters, and the physical plant itself, to ensure that the actual movement of the system is as close as possible to the mathematically generated trajectories. It is very easy to generate a command trajectory that the plant is incapable of following, and this must be guarded against in writing any motion program.

There are two ways to generate parabolic velocity profiles on the parabolic SMCC; 1) directly and 2) through the use of CIRCLE4 mode (also known as S-curve or three-point smoother mode).

First, we will discuss the direct method of specifying these profiles.

8.1.1 Direct Specification

Because there is an extra order in parabolic profiles, the user must specify an additional constraint to define the move. Therefore, in addition to the position and time specifications of a ‘regular’ move, the user must specify the velocity at the end of the move segment. From the constraints of time for the move, end position and velocity that were specified for the move segment, plus starting position and starting velocity, the card will calculate the unique cubic position trajectory necessary to meet the constraints.

The basic command format is as follows:

COMMAND	DESCRIPTION
t1024	Specifies the time for the move
X10000:1024	Specifies the end position and velocity

The small-t time specifier is the same 't' that sets the accel and decel times for non-parabolic moves. For parabolic moves, it sets the time for the entire move segment. One 't' specifier can define the time for more than one move segment. As in other time specification, the units are 'binary milliseconds'(1/1024 sec), so above example specifies a move time of one second. 'T' and 'F' settings do not affect parabolic commands. The line specifying end position and velocity is like any of the Standard Position Move Commands (X, Y, U, V, AX, AY) with a Velocity Command added after a colon. The velocity at the end of the move segment will be:

$$V_{end} = i06 * V_{specified} / 1024 \text{ (cts/sec)}$$

where i06 is the reference feedrate.

In the example above, where the specified velocity is 1024, the velocity generated at the end will be equal to i06. If you wish to specify your velocity directly in cts/sec, simply set i06 to 1024.

NOTE: Methods of changing speed by altering the time base, such as the '%' and 'POT' Commands, or handwheel or pot time base inputs, work independently of this, and can alter the speed from that commanded in these program lines.

WARNING: If the parabolic SMCC leaves a program where the last motion command was a parabolic command with a non-zero velocity, the card will continue to generate a trajectory at this velocity indefinitely. This can lead to an undesired 'runaway' condition. For this reason, the last in a series of parabolic commands should always have a zero velocity Command. Also, one should not try to single-step through the parabolic steps of a program, because the card will continue on at the last commanded velocity between steps, getting completely away from the desired position.

8.1.2 S-Curve (three-point Smoother or CIRCLE4) Mode

The parabolic SMCC provides a special mode that automatically generates and combines parabolic move segments for smooth motion without having the user to specify velocities explicitly. This has several important uses. First, it can provide gentle starting and stopping with S-Curve acceleration, which can greatly reduce the strain on equipment and loads. Second, it can apply a three-point cubic fit to a series of positions received by the parabolic SMCC. This allows the parabolic SMCC to smoothly follow an arbitrary path with far fewer points than would be required if linear interpolation were used. This is particularly useful when working with complex shapes, as in Cam grinding applications, where the number of points around the Cam that need to be specified is an order of magnitude less with three-point smoothing than it would be with straight point-to-point interpolation.

This mode of operation is invoked by using the CIRCLE4 (CIR4) command. Execution of this command automatically cancels the F (feedrate) or T (time move) the parabolic SMCC may have been using. To exit this mode, simply use a new F or T command.

There are three basic rules for generating proper S-curve moves:

1. If you change the time ('t') for a move segment, you must keep the ratio $\Delta P/t$ (change in position/time) the same for the new segment as for the old segment.
2. To change velocity, keep 't' constant, and change the ΔP so that $\Delta P/t$ yields your new desired velocity. The parabolic SMCC will blend smoothly to the new velocity.
3. The parabolic SMCC will automatically add a move segment with a ΔP of zero in front of the first move segment you specify in a series, and also after the last segment. These move segments each take the same time 't' as the move segment to which they are appended (Do not explicitly add these to your program). These segments do not actually cause a zero move; as in rule 2 above, the parabolic SMCC provides a blending to the velocity specified by the next $\Delta P/t$ ratio.

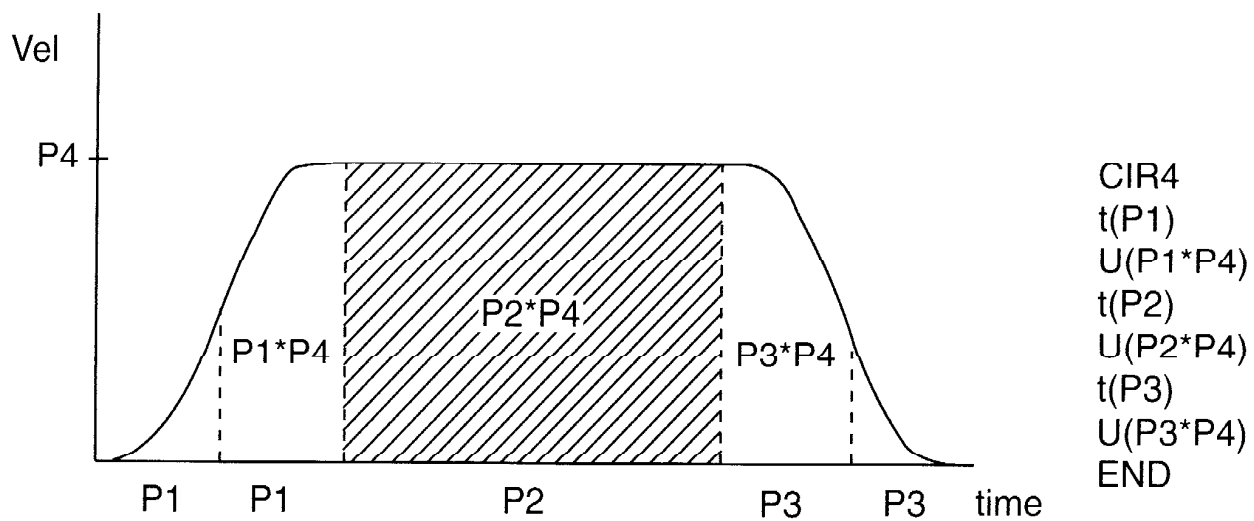
Below is a simple, general-purpose program for generating point-to-point moves with S-curve acceleration and deceleration:

COMMAND	DESCRIPTION
CIR4	Invokes S-curve mode
t(P1)	Move segments take P1 binary milliseconds
U(P1*P4)	Make incremental X-axis move of P1*P4 encoder counts
t(P2)	Time of P2 binary milliseconds
U(P2*P4)	Note that P4 factor keeps $\Delta P/t$ ratio
I(P3)	Constant automatically
U(P3*P4)	
END	End of program

Run velocity = $P4 \text{ counts} / \text{binary millisecond}$
 Total distance = $(P1 + P2 + P3) * P4 \text{ counts}$
 Total time = $2P1 + P2 + 2P3 \text{ binary milliseconds}$

Since the time changes for each segment, the $\Delta P/t$ ratio must remain constant. This ratio is enforced by using the run velocity P4 to multiply by the time to give the ΔP for each segment. The velocity profile generated by this program is shown in figure 8.1.

Figure 8.1 Point-To-Point S-Curve Move



8.1.3 General Algorithm for point-to-point S-Curve Moves

A common type of application for this mode is one that requires simple point-to-point moves with S-Curve acceleration for gentle starts and stops. It is desired to complete these moves in minimum time under the constraints of maximum allowed velocity and acceleration. Below is a general procedure for computing such moves.

First, look at the math for an S-curve acceleration. By symmetry, the area under the curve (the distance covered) is the same as for a straight-line acceleration. If the starting or ending velocity is zero this is just a triangular area, so the formula for distance covered is:

$$\text{deltaP} = (1/2) * 2t * V = V*t$$

where t is the time for each of the two parabolic segments, and V is the non-zero starting or ending velocity.

The top acceleration reached during a S-curve acceleration is exactly twice that of a constant (straight-line) acceleration:

$$A_{\text{peak}} = 2 * V / (2t) = V/t$$

Now if we define:

Vmax = Maximum allowed velocity (cts/sec)

Amax = Maximum allowed acceleration (cts/sec/sec)

then we can compute the time value and distance required for the fastest acceleration to the maximum allowed velocity:

$$t = V / A_{\text{peak}} = V_{\text{max}} / A_{\text{max}}$$

(remember to convert to bin. mSec.)

$$\text{deltaP} = V * t = V_{\text{max}} * t$$

These values can be used for both the acceleration and deceleration portions of the move. For the constant speed (slew) portion, simply calculate:

$$\text{deltaP(slew)} = \text{TotalDistance} - 2 * \text{deltaP(accel)}$$

$$t(\text{slew}) = \text{deltaP(slew)} / V_{\text{max}}$$

Example:

Suppose we had a system with $V_{\text{max}} = 50,000$ cts/sec and $A_{\text{max}} = 500,000$ cts/sec/sec. We wish to do a point-to-point move of 25,000 counts in minimum time with S-curve accel and decel. First, we calculate the accel/decel 't' time as $t = 50000 / 500000 = 0.1$ sec. The accel/decel distance is $\text{deltaP} = 50000 * 0.1 = 5000$ counts.

Next, we compute the slew distance as $\text{deltaP(slew)} = 25000 - 2 * 5000 = 15000$ counts and the slew time as $t = 15000 / 50000 = 0.3$ sec. The program for this move would be:

COMMAND	DESCRIPTION
t102	0.1 sec in binary milliseconds
U5000	Acceleration distance
l307	0.3 sec in binary milliseconds
U15000	Slew distance — note that ratio holds
t102	Decel time = accel time
U5000	Deceleration distance
END	or DWELL — something to ensure stop

Of course, in order to do a three-part S-curve move in this manner, the total distance covered must be enough to allow this acceleration, deceleration, and a short slew. The minimum distance for which this is possible is:

$$\text{deltaP}(\text{min3part}) = 2 * V_{\text{max}} * t + 0.020 * V_{\text{max}} \\ 2 * (V_{\text{max}}^2) / A_{\text{max}} + 0.020 * V_{\text{max}}$$

For the above example, this distance would be 11,000 counts. If the distance is shorter than this, it will be necessary to do a single part S-curve move.

For a very short move, the profile will never reach maximum allowed velocity, so maximum allowed acceleration is your only constraint for calculating a minimum-time move.

$$\text{Minimum-time } t = \sqrt{\text{deltaP} / A_{\text{max}}}$$

Continue from above example, a move of 5000 counts is too short for a three-part move, would have

$$t = \sqrt{5000 / 500000} = 0.1 \text{ Sec.}$$

The program would be:

COMMAND	DESCRIPTION
t102	0.1 sec — total move time = 0.3 sec.
U50000	Incremental move total distance or
END	Dwell — ensure stop.

Another type of application for this mode is in providing irregular curve following capability while smoothing the trajectory between adjacent points on the curve. The parabolic SMCC simply requires a series of positions and the time to move between the positions. The parabolic SMCC, once in CIRCLE4 mode, takes each data point in turn, and does a cubic fit with it, the point before it, and the point after it called three-point smoothing. This process is done in real time, and it is exactly the same process as described above.

This smoothing may take the calculated path slightly away from the actual specified points, particularly if the velocity changes are sudden (because of the blending action). But generally, this mode allows the motion system to follow the desired position profile very closely with far fewer points than with straight-line interpolation, and most importantly, it does so without any sudden acceleration changes ('jerk') at point boundaries that could excite undesirable system dynamics.

For example, this mode is very useful for retracing a shape that has been digitized. The time 't' between digitized points would be specified, and the parabolic SMCC would follow the path between points, passing each new one at time 't' past the previous one.

For this type of application, we look at the equations that derive the derived commanded positions from the numbers in the program (to repeat, the math in the parabolic SMCC is the same as in S-curve type applications; this is just a different way of looking at it). For a series of programmed positions $r(n)$:

$$\text{3-point derived pos } (n) = [r(n-1) + 4*r(n) + r(n+1)] / 6$$

$$\text{3-point derived vel } (n) = [r(n+1) - r(n-1)] / 2t$$

Consider the following short program to fit a series of points:

COMMAND	DESCRIPTION
X=0	Define the current position to be zero
CIR4	3-point smoothing mode
t102	Time between points is 0.1 sec.
X1000	Move to X=1000 (deltaP = 1000)
X2000	Move to X=2000 (deltaP = 1000)
X2500	Move to X=2500 (deltaP = 500)
END	End of the program

The parabolic SMCC will automatically add zero-distance moves at the beginning and end of the sequence (rule 3). This makes the move data sequence:

0, 0, 1000, 2000, 2500, 2500

The derived positions and velocities at time intervals 't' are:

- 1 $p(1) = [0 + 4*0 + 1000]/6 = 167$ cts
 $v(1) = [1000 - 0]/[2*0.1] = 5000$ cts/sec
- 2 $p(2) = [0 + 4*1000 + 2000]/6 = 1000$ cts
 $v(2) = [2000 - 0]/[2*0.1] = 10000$ cts/sec
- 3 $p(3) = [1000 + 4*2000 + 2500]/6 = 1917$ cts
 $v(3) = [2500 - 1000]/[2*0.1] = 7500$ cts/sec
- 4 $p(4) = [2000 + 4*2500 + 2500]/6 = 2417$ cts
 $v(4) = [2500 - 2000]/[2*0.1] = 2500$ cts/sec
- 5 $p(5) = [2500 + 4*2500 + 2500]/6 = 2500$ cts
 $v(5) = [2500 - 2500]/[2*0.1] = 0$ cts/sec

As explained before, the 't' time does not need to remain constant, but if it is changed, the ratio of deltaP to 't' must stay fixed (rule 1), or there will be discontinuities in velocity and acceleration, defeating the purpose of this mode of operation.

8.2 1/T ENCODER PERIOD

Normally the SMCC measures the number of encoder counts received during each servo cycle. At low velocities the number of encoder counts received per servo cycle is zero or very low, causing granularity and errors in actual velocity calculations. The parabolic SMCC provides the digital tachometer (1/T) mode which uses time measurement of position changes for effective fractional resolution. The time measurement uses a 2 MHz clock and the 16-bit handwheel encoder counter to accumulate the time elapsed from one count to the next; this provides an accuracy of ± 0.5 microseconds and a maximum range of 32 milliseconds between pulses. The result of using 1/T mode is that low velocity performance and smoothness is improved by at least a factor of two or three, which allows higher gains to be used.

Since there is system dependence on the number of encoder counts per revolution for low velocity smoothness, good general rules are:

- A) Use higher encoder count/revolution while keeping in mind that the high speed requirements should not exceed the maximum count rate of the encoder or the controller's position counts.
- B) Use at least 100 to 200 counts/second at the lowest RPM that you will use to avoid velocity granularity.
- C) Without 1/T, use at least 300 counts/second for equivalent operation.

Lower encoder counts/second may be used if more velocity ripple can be tolerated or servo gain can be reduced for smoother velocity at the expense of stiffness.

i28[i48] is used to enable the 1/T mode and jumpers E40 and E41 must be removed so the handwheel counters can be used by the 1/T mode. Handwheel inputs are disabled and cannot be used when in 1/T mode.

8.3 DATA GATHERING

The parabolic SMCC has the ability to collect and store servo data in real time (position and following). This can be used to provide a "snapshot" of system performance during a period of interest. The servo data can be stored in any section of the card's memory space (it must be a continuous section of memory). The data can be sent to a host computer for further processing (The SMCC Monitor Program automates the data gathering process).

The fundamental parabolic SMCC command to set up the data gathering is:

y f , b , a , d1 , d2

where

y: is the command itself;

f: defines the sampling period (f is the log [base 2] of the sampling period in servo cycles)

b: is the number of bytes stored each cycle (1 – 32);

a: is the address of the first byte to be collected each cycle (default is fe48 : X position);

d1: is the starting address of the storage buffer (default is the address following the program buffer);

d2: is the ending address of the storage buffer (default is the end of the open memory).

On receiving this command, the parabolic SMCC will prepare for data storage and return the address of the pointer to the data storage buffer (in hex).

Note: Sending the 'y' command without any arguments will do nothing but return the address of the pointer. The host should send a 'y' command without arguments to get the starting address.

The program in the SMCC must set and clear in the fifth bit of the byte at address febd to start and stop the data gathering. This is usually done in the following manner:

SET C20 = febd
SET M20 = C20.5
...
(instructions before gathering)
...
SET20
...
(instructions while gathering)
...
RESET20
(instructions after gathering)

After the data collection has been completed, the host should send a 'y' command without any arguments to get the address (End address). Then the 'd' command is sent, with two arguments: first is the starting address (in hex), and the second is the length of the buffer (end address minus starting address) in decimal. You will need a host program (such as the Monitor program) to translate the internal form (HEX table) into a physical value.

The number of bytes to be stored in a servo cycle is defined by the type of that data. This is further explained as follows:

BYTES TO COLLECT IN A GIVEN SERVO CYCLE	DATA TO COLLECT
4 bytes	X Position
8 bytes	X position / Y Position
12 bytes	X position / X Following Error
16 bytes	X & Y Position / X & Y Following Error

If 12 byte format is adopted, only the first 4 byte and last 4 bytes is used to store X position and X following error. The velocity and acceleration can be derived from the positional data and given servo cycles.

8.4 DMA COMMUNICATION

The parabolic SMCC provides a DMA (Direct Memory Access) communications mode in order to allow rapid transfer of data to and from the host computer. Parameter i19 (Address of last card, default value is "0", range "0" to "f") is used by the cards to receive and send "packet" data at high rates. Card addresses need not be consecutive but the "empty slots" will waste communications time.

Communications data consist of parabolic commands (with position and velocity) or non-parabolic commands (position only) and CIR4 three-point smoothed data mode. To select parabolic (position velocity) or standard (position only) mode, use the i63 parameter as follows:

i63 0 Selects the parabolic mode

i63 1 Selects the standard mode

Note that position or status data sent from the parabolic SMCC to host are not affected by i63.

A. DMA, Host to Parabolic SMCC

Data can be transmitted via RS232 or parallel port by the host and may be either placed in the program buffer of the parabolic SMCC or be executed on-line by placing the parabolic SMCC in the MDI mode (i02 set to 1).

To start a DMA transmission assuming parabolic position mode: position and velocity data (i63=0), the host must first send a <CNTRL B> character (02H) followed by the move data for card A0 which consists of 14 bytes of data per card and represents:

4 Bytes of X Position Data

3 Bytes of X Velocity Data

4 Bytes of Y Position Data

3 Bytes of Y Velocity Data

Data format is binary LSB first, MSB last. The position data for both X and Y must be padded with five zeros in the LSB word (32 times greater) since the actual position data has 27 bits of data only. Upon completion, data for A0, data for A1, and so on is transmitted as required. The transmission is terminated when all the data has been transmitted. If the MDI mode is used (i02 set to 1), then the motion will begin upon reception of the data. If i02 is set to zero (buffer mode), the data will be placed in the buffer and may later be executed by Step "S" or Run "R" Commands. Data transmission to the card in the buffer mode may be mixed with both binary(DMA) and ASCII (non-DMA) data.

The total transmission in i63=0 mode consists of:

<CNTRLB> + (14 Bytes x Number of Cards)

In i63=1 mode, standard data with position only, velocity data is not sent, and the total transmission will consist of:

<CNTRL B> + (8 Bytes x Number of Cards)

The data transmission rate with parallel port is approximately five microseconds per character. RS232 transmissions occur at the selected baud rate, the fastest of which is 38,400 baud or approximately 250 microseconds per character. These rates apply to both reception and transmission of data. All DMA transmissions generate an internal interrupt with a latency of approximately eight microseconds.

B. DMA, Binary Parabolic SMCC, to Host

To receive position data back from the parabolic SMCC, the host computer issues a <CNTRL C> (03H) character. Upon reception of <CNTRL C>, the A0 card immediately begins transmitting binary data consisting of four bytes of X position data and four bytes of Y position data, LSB first, MSB last. The SMCC in DMA mode generates an interrupt request internally which overrides the servo cycle in order to send data back to the host. The time needed to gather the data and begin transmission to the host is approximately eight microseconds. Each of the four bytes has five zeros padding the LSB byte so that the data is 32 times larger than the actual position data.

The host must receive the data and after eight bytes of data have been received, it also must issue a "space" character (or any other character) to cause card A1 to transmit its eight bytes of data. It must continue to do this until all cards have transmitted their data.

To request status data back from parabolic SMCC the host computer issues a <CNTRL D> (04H) character. Upon reception of <CNTRL D>, the A0 card immediately begins transmission of its eight bytes of status words. The host then sends a "space" or any other character back to the parabolic SMCC in order to receive eight status bytes from the next card. It continues until all cards have sent their status back. See next Paragraph for the contents of the status word.

The difference in transmission of status between the DMA binary mode and the ASCII mode is that in ASCII each byte of status is sent as two 4 bit nibbles each; in hex ASCII format with most significant nibble first. Total status transmission is therefore 16 characters in ASCII mode. Also, the time needed between reception of command and transmission of data is in the order of four milliseconds and in addition each card must be addressed individually and then requested to send data back with the "?" command.

Appendix A

Glossary of Initialization/Setup Commands

This appendix is dedicated for the SMCC setup for DC motors. Certain controls (such as Baldor's BSC brushless control) have a built-in commutation function and act as a simulated DC motor. In these cases, the setup parameters introduced in this appendix still apply.

Initialization commands and associated arguments (Setup parameters) are used to configure the SMCC operation conditions. Sending the command "inn" without any parameter value is interpreted as a parameter value query, the SMCC prompts with the value of the stored parameter. Where nn can be any number for 00 to 81 (depends on the version number). Sending an "ix" command cause the SMCC to send all parameters at one time.

Example A.1

COMMAND	DESCRIPTION
i00	Query to check the following error fault limit. The SMCC will respond with the currently stored values.
i00 2000	The new parameter value replaces the old value.
i002000	Same as above.
I00 2000	Buffer command - will be processed at the time of program execution.

For quick referencing, the initialization commands are introduced according to their numeric order, i00, i01 .. etc. The equivalent Y axis commands are those embedded in [].

i00 Set the following error Limit (FAULT)

Range (0 +/- 32767), Default (2000), Unit: Encoder Count

The SMCC keeps track of the following error (Positional error between the desired position and the actual position), If a positive limit is used and the servo cannot follow the desired position within the specified limit, the SMCC turns off the control and the GLT (Get Lost Timer) is set, thus disabling the SMCC and outputting to the FEOD/. This is a fatal error and the operator has to reset the system to resume operation. The FEOD/ output may be used to turn the system's power off.

If a negative limit is used in i00, the following error is still detected and the control disabled, but the SMCC remains active and the host can communicate with it, even though it has stopped motion. A soft "\$" RESET or hard "\$\$" RESET may then be used to clear the fault and re-enable the SMCC.

To disable the following error fault, set i00 to 0.

i01 Set the in-position Band

Range (0 - 65535), Default (100) Unit: Encoder Count

When the SMCC is at rest, it turns on the in-position light and the IPOS/ output whenever the actual position is within the specified counts from its present desired position. Once in-position has occurred, the SMCC is ready for the next move. IPOS will first occur after a move is complete and the SMCC has reached its actual desired position within $\pm i01$ counts. Not when approaching position within the IPOS band. Subsequently, IPOS will remain true as long as the actual position stays within this band and before the next move commences.

i02 Set the Manual Data Input Mode

Range (0,1), Default (0), Unit: None

This parameter decides whether motion buffer commands will be stored in the SMCC memory for later "RUN" (Default) or processed as soon as it is received.

- 0: Reset MDI Mode (Default) Motion buffer commands are stored in the motion buffer and can then be processed by the "S" (Step) or "R" (Run) command or by the "Step" or "Run" front panel commands.
- 1: Set MDI Mode Buffer commands are immediately processed upon reception of a <CR> character. This mode is used to execute one command at a time without using the motion buffer, i.e., to test-run programs.

Note that the processing of direct control commands (Non motion buffer commands) is not affected by this mode. Continuous motion can be achieved in this mode by issuing a second command while the first is being executed. To exit the MDI mode, i020 may be entered or a "S" (Step) or "R" (Run) command may be given.

When in MDI mode, a "F" (Feedrate) or "T" (Time) must be issued at least once in order to cause the SMCC to make a move when it receives a motion command. Exercise caution because if returning to the program flow, the latest feedrate will be in effect until a new feedrate command is executed by the program.

i03 Set the Handshake Enable in Host Communication

Range (0,3), Default (1), Unit: None

- 0: Handshake is disabled. The processor does not send handshake messages. But will respond upon reception of query commands.
- 1: Handshake is enabled. The processor sends a <LF> character upon reception of a <CR>, and sends a "BELL" upon unrecognizable commands.

-
- 2: Handshake is enabled; responds with an <ACK> for valid commands or a<BELL> for illegal commands. Messages are sent without a <LF>.

Data is sent as: DATA, <CR>, (DATA, <CR>, . . .) <ACK> (with an <ACK> sent after the last line only).

- 3: Handshake is enabled; sends an <ACK> or a <BELL> as appropriate. Data is preceded by a <LF> and followed by an <ACK>. Data is sent as: <LF>, DATA, <CR>, (<LF>, DATA, <CR>, . . .) with an <ACK> (after the last line only).

NOTE: Regardless of the handshake, the SMCC sends messages if a command specifically asks for it (as in f <send following error>, inn <send stored inn parameter value>, etc.)

i04 Set the Home Feedrate

Range (0 - 255), Default (0), Unit: Encoder counts/second.

This parameter sets up the speed of the homing operation with respect to the reference speed set by i06. Home feedrate is calculated by the following equation:

$$\text{Home feedrate} = (i04 / 256) * i06$$

If i04 is set to zero, the homing function will not occur. Refer to i24[i44], i25[i45] and i26[i46].

i05 Set the Servo Time Period

Range (0 - 65535), Default (487), Unit: Microsecond

This parameter specifies the period of each servo cycle and is calculated by:

$$\text{Servo Time Period} = 2 * (i05 + 1) \text{ in micro Seconds}$$

Caution should be exercised when changing i05 since i20, i21 and i23 are affected and must be readjusted; i20 (proportional gain) may be increased if the servo cycle is shortened.

Note: i05 should not be changed under normal conditions. If the servo time period is shortened too much, there will be insufficient time to handle the communications and the PLC and may cause the SMCC to hang up.

i06 Set the Reference (Maximum) Feedrate

Range (0 - 8388607), Default (2000), Unit: EC/sec.

This parameter specifies the reference feedrate so that the commands i31[i51] Jog Feed Rate, i04 Home Feed Rate, .. etc. can be specified as a ratio.

Note that the actual machine feedrate is also a function of the “F” (Feedrate) command:

Feedrate = (nn / 1024) * i06 (Encoder counts/Sec.) and

Jog speed = (i31[i51]/256) * i06

Home Feedrate = (i04/256) * i06

The following conditions will be achieved:

100% feedrate when you enter the F1024 command.

Maximum home feedrate when i04 is set as 255

Maximum jog rate when i31[51] is set as 255

i07 Set the Time Scale

Range (0 - 65535), Default (65498), Unit: None

This is a system software related constant. It is recommended that the user set this value from the following equation if the servo time (i05) is changed.

$$i07 = 65536 * (2 * (i05 + 1)/dT)$$

Where dT = the units for the T or t commands (in microseconds)

If dT = 1/1024 sec. and i05 = 487 then i07 = 65498. This is the default value.

i07 serves as the time base upon power-on or reset. Once a “%” or “POT” command is executed, the value with that command overrides the i07 value for moves. However i07 always control the time base for the DWELL commands.

If i05 is changed, in order to speed up or slow down the servo cycle, i07 should be readjusted in order to keep the actual time scale of the SMCC constant. The standard time scale is dT = 1/1024 sec. (one binary millisecond). Also note that if i07 is changed, the on-line (non-buffer) % feedrate command (POT in buffer) must also be readjusted.

NOTE: i07 should not be changed in normal condition.

i08 Set the Acceleration/Deceleration Time

Range (0 - 65535), Default (512), Unit: 1/1024 sec.

This parameter governs the total accel/decel time whenever speed change occurs. When i08 = 1024, the accel/decel time is one second.

The SMCC uses a time-specified accel/decel format instead of a rate specified format. The user may specify the accel/decel time before every move.

The accel/decel time is also controlled by the "t" command in the program buffer. The accel and decel times are equal and both controlled simultaneously. It is possible to make the accel and decel different by breaking up one move into two consecutive moves with an I08 or "t" change between the split blocks.

If i11 = 1 (feed hold slew control on), then i08 sets the deceleration time upon hitting a software or a hardware limit (on either a jog or a calculated move), or for after an H (feed hold) command on a jog move.

i08 is read once upon power turnon or reset and sets the value of "t" initially. Once a "t" command is executed, i08 does not affect the calculated (programmed motion) accel/decel, but only the jog accel/decel. It is therefore possible to have different accel/decel values set for jog and programmed moves.

i09 Set the Position Integration Mode

Range (0,1), Default (1), Unit: None

To reduce the following error, the SMCC utilizes a servo controller which includes position integral feedback. i09 decides whether this integral feedback is activated during the move or not. Caution must be exercised since this mode may result in consistent overshoot when stopping. See i23[i43] for integrator time constant.

- 1: Position integration is enabled only when the motion is stopped. The integrator is automatically activated at the end of the move when velocity becomes zero.
- 0: Position integration is enabled all the time, both during a move and when stopped.

i10 Set the Limit for the Following Error Indicator

Range (0 - 65535), Default (1000), Unit: Encoder Count

If the following error is greater than this limit, the SMCC turns on the following error light. However, shutdown does not occur unless the i00 limit is reached. This parameter is useful in providing a visual indication when the following error output is tied to an LED.

The FILD/ signal may be tied to FEOP/ by installing the E22 jumper. This causes both the i00 and i10 to generate FEOP/.

i11 Set the Feed Hold Slew Control

Range (0,1), Default (1), Unit: None

This parameter determines the rate of deceleration to a stop.

-
- 0: Feed hold slew control is disabled. When feed hold occurs, the feedrate change is accomplished as rapidly as possible without controlled deceleration. The motor will actually come to a stop and then return to the point at which feed hold occurred.
 - 1: Feed hold slew control is enabled. The deceleration rate is controlled by i08 or the % slew rate if in a calculated move. The motor comes to a stop at the end of decel. The % slew rate value is deducted every servo cycle from the current setting of the % command until a zero value (stop) is reached. The deceleration rate is controlled by i08 if a hardware or software limit is hit, or if a "q" command is given.

i12 Move Calculation Time

Range (0 - 65535), Default (40), Unit: Binary Milliseconds,
Minimum (10)

The Parabolic SMCC uses i12 as a predefined servo calculation time to ensure that in multiple-axis systems, all cards will begin motion simultaneously. A certain amount of time is always necessary to do the calculations. It may become necessary to use a higher value of calculation time when moves based on variable parameters, especially those involving divisions.

When the SMCC is making a normal trapezoidal move, the calculation for the next move begins i12 time before the end of the move, which is the beginning of decel ("t" time is the accel/decel time).

When the SMCC makes a parabolic move, i12 time occurs at the very end of the move because "t" time represents the move time and a parabolic move does not have a defined deceleration time. If i12 time becomes longer than "t" time in either trapezoidal or parabolic moves, calculations are then made in "t" time instead of i12 time.

i13 Select the Motor Type

Range (0,1), Default (0) Unit: None

- 0: Select brush type DC motor control. Two axis control is possible. (See i15)
- 1: The SMCC will be configured such that it commutates the 3-phase motor current of a brushless 3-phase drive. This is the proper setting for the SBTS 10, SBTS 20, and SBSC drives which both utilize the μ SMCC internally.

All other Baldor DC and AC drives where the commutation sequencing is done at the motor or brushless drives that commute without the SMCC should have i13 set to zero. The drives include: PMC, LD, SUM, THM, and the SBTS 15.

The two error output signals from the SMCC now function as independent ± 10 volt DC outputs for the X and Y axis.

NOTE: When i13 is set to 1, the function of i40 – i59 is changed to control the setup of the motor commutation. Use the definitions that start on page A-21.

i14 Select the Processor A/D Converter Mode

Range (0,1,2,3), Default (0), Unit: None

There are four A/D channels available on the SMCC, two channels [/ANA2(pin J1 – C1 for the X axis) and ANA3 (pin J1 – A1 for the Y axis)] are controlled by i14 as follows:

Note that regardless of the mode selected, the ANA0 and ANA1 channels are activated to perform the overload protection.

0: A/D converter mode disabled.

The analog inputs ANA2 and ANA3 are not used. (Default)

1: Feed pot mode (same as the % command)

The SMCC uses the ANA2 input to control the feedrate (ANA3 is not used). This allows a simple variable voltage input (0 to +10V DC) through a potentiometer to be used as the program execution rate control.

This is just like rate controlled by the % command.

Do not try to control multiple cards with an analog input; the A/D converters on the different cards can not be counted on to give identical results.

Proper shielding of the analog input wiring is very important in this mode. Noise on the line will translate into very noticeable movement.

Do not have i16 = 2 when i14 = 1, because with such a setting, the handwheel input will override the analog input voltage in controlling the feedrate.

2: Jog speed control mode

In this mode, the analog inputs control the speed of the jog moves: ANA2 controls the X-axis speed; and ANA3 controls the Y-axis speed. The analog inputs thus replace the function of the jog speed parameters i31 and i51. The starting and stopping (and direction) of the jog moves are still controlled by the jog commands (J, j, JY, jY).

3: Velocity control

Velocity is controlled by a variable voltage input (+/- 10V) through a potentiometer, and is bi-directional with 4 speed ranges controlled by the XJPL (X axis jog positive, J1 – B30), and XJMI (X axis jog negative, J1– B32) [J1 – B39 and J1 – B31 for the Y axis] inputs. This will increase the relative resolution for low speed operation.

Jog Positive Input	Jog Negative Input	Speed Select
Inactive	Inactive	x 1
Active	Inactive	x 2
Inactive	Active	x 4
Active	Active	x 8

The remaining two A/D channels are used to monitor the motor current and provide I²t protection.

i15 Set the Y Axis Disable in DC Motor Mode (i13=0)

Range (0,1), Default (1), Unit: None

0: The Y axis is enabled.

1: The Y axis is disabled.

If a single DC motor is to be controlled, connect it to “X” axis, and configure i15 as “0” to block the “Y” axis logic process. If the Y axis is disabled, the “Y” error output can be used as a programmable voltage generator by using the OYn command.

NOTE: In a single axis situation, the limit switches for the Y axis must be disabled (grounded); otherwise the SMCC will not make any moves.

i16 Select the Handwheel Operating Mode

Range (0,1,2,3), Default(0), Unit: None

0: Standard mode — The handwheel input is used to change the motor's position. The number of pulses represents the position and the pulse frequency represents velocity. The ratio of the handwheel to the motor movement or “scale factor” is governed by i27[i47] (where a value of 4096 for i27[i47] represents a scale factor of one) and the mode of operation by i28[i48].

1: Leader follower mode (Master slave) — Moves strictly follow handwheel encoder input. This mode is like mode 0 except for the fact that calculated moves are not possible. This is a special mode allowing a very fast servo cycle thus permitting “tight” handwheel following. The scaling is governed by i27[i47] and handwheel mode control is governed by i28[i48].

2: Synchronization mode — Time base uses the Y handwheel. Externally injected pulses or quadrature encoder inputs can be used to control the SMCC's time base so that multiple cards will use an identical time base (variable) for synchronous program execution. The program execution rate is then slaved to the external clock or encoder frequency.

$\text{Time Base} = \text{Input frequency} \times \text{Mult.Factor} \times \text{ScaleFactor}$

A HW encoder rate of 65498 Hz, hardware multiplied by 4 on the SMCC and scale factor multiplied by 16 (i27[i47] = 65535) results in a 100% feed rate execution of the program.

3: Time base uses the X handwheel; otherwise the same as mode 2.

NOTES: When using i16 mode 2 or mode 3, it is a good practice to disable the X or Y axis by setting i28[i48] to 3. (Do not set i48 to 3 on the μ SMCC)

When you set i16 to mode 2 or 3, do not have i14=1, or you may have a conflict as to what is controlling the execution speed.

Also refer to i27[i47].

i17 Set the Velocity Display Time Base

Range (1 - 2^{31}), Default (30000), Unit microseconds

i17 sets the sample period at which encoder counts are measured and converted to velocity.

i17 = "sample period" in microseconds

The data displayed for velocity is the distance that the SMCC measures, in scaled encoder counts (see i38[i58]), during the "sample period" specified by i17.

The velocity is measured and displayed at the rate of approximately three times per second. By using the i38[i58] X and Y scale factor, it is possible to adjust the scaling so that on systems that have different encoder counts per motor revolution for X and Y, the displayed velocity can be made to be identical.

i18 Select the Display Type to be Used

Range (0 - 3), Default (3)

This parameter selects the type of hardware output display that the SMCC will address. The display information can include position, velocity, following error, and status bytes for each axis. The settings are:

0: For an LED display: 2 units x 1 line (8 digits/line)

The 1st display shows the X axis' position.

The 2nd display shows the Y axis' position if i13 = 0;

1: For an LCD display: 2 units x 2 line (24 digits/line)

The 1st display shows the X axis' position and the X axis' velocity on the 1st line and the X axis' F.E. and status on the 2nd line.

The 2nd display shows the Y axis' position and the Y axis' velocity on the 1st line and the Y axis' F.E. and status on the 2nd line.

2: For an LCD display: 2 units x 1 line (40 digits/line)

The 1st display shows the X axis' position, velocity, F.E., and status;

The 2nd display shows the Y axis' position, velocity, F.E., and status.

3: For an LCD display: 1 unit x 2 lines (40 digits/line)

The 1st display shows the X axis' position, velocity, F.E., and status on the 1st line; and the Y axis' position, velocity, F.E., and status on the 2nd line.

The 2nd display can show the X axis' F.E. and status on the 1st line; and the Y axis' F.E. and status on the 2nd line; useful if the actual displays are shorter than 40 digits long.

The X axis' position format is governed by i38.

The Y axis' position format is governed by i58.

The X axis' velocity format is governed by i17 and i38.

The Y axis' velocity format is governed by i17 and i58.

The X axis' following error format is governed by i38.

The Y axis' following error format is governed by i58.

The X axis' status shows the bits of the 3rd status byte, feb7 for Parabolic SMCC)

The Y axis' status shows the bits of the 4th status byte. feb8 for Parabolic SMCC)

Refer to the SMCC memory bit map in appendix D for definitions of the individual bits.

If you are not using a display, assigning i18 to 0 will decrease the background time and cut the interruptions to the other background tasks.

i19 Set the Highest Card Address

Range (0 - 15), Default (0)

i19 is used to set the highest card address in a multiple card the system. The value of i19 is used in DMA mode to allow the SMCC to receive and send "Packet" data without interruption for card addressing. It is not necessary to have all card addresses consecutive to each other. However, if there are gaps, the host would have to send extraneous data to non-existing cards. DMA is automatically received by each card in succession and it is also sent automatically in successive order. DMA mode allows only binary data transmission of position and velocity and can be entered by sending the characters <CNTRL B, C, or D>.

i20[i40] Set the Proportional Gain Constant (if i13 = 0)

Range (0 - 65535), Default (50), Unit: None

This is the major gain constant in the software oriented PID controller. The SMCC produces an error correction signal which increases, in direct proportion, to the amount of following error that is observed. The output is as follows:

$$(1.25 * i20[i40])/65536 = \text{VOLTS/ENCODER COUNT}$$

Usually i20 is proportional to the "stiffness" of the loop.

The SMCC's servo algorithm is shown here for reference.

$$\begin{aligned} E_{out} = & i20 \cdot k0 \cdot \{EP(n) - [i21 \cdot AV(n)] + i22 \cdot DV(n) \\ & + i30 \cdot k1 \cdot [DV(n) - DV(n-1)] + i23 \cdot k2 \cdot IP(n)\} \end{aligned}$$

Where:

i20 = Proportional gain

k0 = 1.25 volts/65536
(Used to scale the proportional gain)

EP(n) = Position error = Dpos(n) – Apos(n)

Dpos(n) = Present desired position

Apos(n) = Present actual position

i21 = Differential gain (Velocity feedback)

AV(n) = Present velocity = Apos(n) – Apos(n – 1)

i22 = Velocity feedforward gain

DV(n) = Desired velocity = Dpos(n) – Dpos(n – 1)

i30 = Acceleration feedforward gain

k1 = 4 (Used to scale the accel feedforward)

i23 = Integral gain

k2 = 1/4096 (Used to scale the integral gain)
n – 1

IP(n) = SUM EP(j) (integral of the position error)
j = 0

(n–j) is used to represent the value j servo cycles old.

NOTE: The actual velocity is modified when the 1/T mode (chapter 8) is used by the ratio of the servo time to the actual measured time between encoder counts.

i21[i41] Set the Differential Gain Constant (Feedback Term) (If i13 = 0)

Range (0 - 255), Default 40, Unit: None

The differential gain constant is used as the velocity feedback to stabilize the servo loop. The greater the differential gain, the more stable the system is, but also the greater the following error (This following error can be counteracted using i22[i42] velocity feedforward).

Increasing differential gain can suppress system overshoot and oscillation.

i22[i42] Set the Velocity Feedforward Gain (If i13 = 0)

Range (0 - 255), Default (0), Unit: None

i22[i42] is used to generate a term proportional to the desired velocity that is added into the servo command. This term is very useful in reducing the following errors caused by differential gain damping (in systems with no tachometer). The damping is necessary for system stability, but it introduces a following error roughly proportional to the velocity. Velocity feedforward can then be used to eliminate (or greatly reduce) this following error, and since it is not part of the feedback, it does not affect the system's stability. Also, it is more useful than error integration for reducing the following error on variable-speed moves, because error integration cannot keep up with the varying speeds as feedforward does.

In a system without a tachometer, i22 should generally be set equal to i21.

i23[i43] Set the Integral Gain Constant (If i13 = 0)

Range (0 - 255), Default (0), Unit: None

The integral gain of the SMCC is used to correct the residual position error due to off-balanced loads, friction, temperature, and time drift, etc. It may also be used to dynamically reduce the following error while the SMCC is performing a move.

The integrator makes a correction every servo cycle and the amount of correction is controlled by the value of i23[i43]. The "Capture Range" of the integrator is 1000 encoder counts. This means that the SMCC will correct up to that much error automatically. Greater errors will leave a residual error.

Parameter i09 controls when the integrator will be active. When i09 = 1, i23[i43] takes effect, but only when the SMCC is at rest; when i09 = 0, i23[i43] is effective at all times.

i24[i44] Select the Home Direction (If i13 = 0)

Range (0,1), Default (0)

0: Home in the negative direction.

1: Home in the positive direction.

This parameter sets the direction in which home will take place. (See i04 for homing feedrate setup and i25[i45] for home offset). Note that during a homing procedure, the i35 [i55] and i36[i56] software travel limits are disabled.

i25[i45] Set the Home Offset (If i13 = 0)

Range (+/-32767), Default (500), Unit: EC

During the homing process and after detection of the machine's home switch (Home flag) and the encoder's "C" channel, the axis will move by the amount and direction specified by i25[i45]. (See i26[i46] for definition of polarities that may be used for the "C" and home flag inputs). Note that the "User flag" input may be used as well. i35[i55] and i36[i56] software travel limits are disabled during the homing procedure.

i26[i46] Set the Home Flag and the User Flag Control Bits (If i13 = 0)

Range (0 - 7), Default (0), Unit: None

The SMCC can receive interrupts from several different sources based on the hardware configuration. i26 configures this function so that an interrupt will cause the SMCC to jump to the correct process.

To interpret the following logical convention, A "/" after the signal name means the logical complement, not the differential signal; "AND" denotes the formal Boolean operator.

i26[i46] Value	Triggering Condition
0	(C/ and HF/) falling edge
1	(C and HF/) falling edge
2	HF rising edge
3	UF rising edge
4	(C/ and HF) falling edge
5	(C and HF) falling edge
6	HF falling edge
7	UF falling edge

The combination of C channel and Home Flag is useful because it allows the home switch to be located loosely, with the C channel providing the precise position.

i26 also establishes the condition of the X-axis inputs that generate the trigger in the "Go Until" move ($X_d \wedge d$). The X-axis interrupt acts on both axes for this situation only. i46 establishes the condition of the Y-axis inputs that generate the trigger for the WUT (Wait Until Trigger) moves.

i27[i47] Set the Handwheel Scale Factor (If i13 = 0)

Range (0 - 65535), Default (4096), Unit: None

This determines the ratio of handwheel counts vs. motor encoder counts. The desired position indicated by the handwheel is calculated as follows.

$$\text{Desired Position} = (i27[i47] / 4096) * \text{Handwheel Count}$$

The maximum ratio yields 16 times the input count. Exercise caution when using a high ratio.

i27 = 4096 sets a one-to-one ratio between the two values.

When i16 is configured as mode 2 or mode 3, the program execution rate is slaved to the external clock or encoder frequency.

The external frequency is used by the SMCC to calculate the effective “%” it will use when it makes its moves. Calculations are made every servo cycle and the “SLEW” control (rate of change of “%”) is disabled in order to avoid errors when multiple SMCCs are used. The result of instantaneous changes in “%” may have some velocity granularity. It is therefore recommended that a higher frequency be used for the handwheel time base. This reduces the granularity quite effectively. Caution must be exercised to limit the maximum input frequency. Select an input frequency that will not cause the % command to overflow (The maximum value for the % command is 65535. When an overflow occurs, the SMCCs speed can drop to 0 or a small value, since %65536 is the same as %0.)

Another way to discuss the control of the SMCC's % command is to consider that with a % value of 65,498, a T of 1024 would represent 1 second. If the % is changed to 1/2 of its value, 32749, a T of 1024 will represent 2 seconds. The % command therefore controls the time base of the SMCC.

A programming example is described for a machine with a spindle driver and two traverse axes which will wind wire on the spindle. The system's physical data is as follows:

Example A.2

Spindle: Max. RPM = 600 = 10 RPS.

Total travel = 100 Revolutions

Encoder Counts/Rev = 2000 X 4 = 8000

This is an independently driven axis.

Traverse: The Y axis is expected to vary its pitch every 10th rotation of the spindle. The total travel is 10 inches; 10 rotations of the spindle should cause the traverse to move 1 inch. The X axis moves across at a constant pitch. Both the X and Y axes are “slaved” to the spindle axis and the spindle axis encoder is fed to the SMCC's “X Handwheel” input and is used to establish the “%” value which controls the SMCC's time base.

COMMAND	DESCRIPTION
A0	Address the card
z100	Clear the buffer.
i283	Disable the X and Y handwheel modes
i483	Assume 8000 counts/rev. of spindle
i163	The X axis handwheel is the time base for the SMCC
i27100	Make 1 binary mSec. = 100 counts
i4732768	100% feedrate (%) = counts/mSec.=(i27 x 32768)/i47 = 100, use i27 and i47 for more flexibility and precision 1 rev = 80 mSecs. of time = 80 x 100 = 8000 counts of the spindle 100 count/mSec. is %65498 = 100% feedrate. The actual time will be 1 Sec. per 10 revs. 600RPM total move, 10 seconds, 80000 counts.
T800	Time for 10 revolutions of the spindle
U(P1)V(P2)	10 revs. P1 and P2 are the distance to move
U(P1)V(P2)	20 revs, change ratio (Y axis)
U(P1)V(P3)	30 revs, change ratio (Y axis)
U(P1)V(P3)	40 revs, change ratio (Y axis)
U(P1)V(P4)	50 revs, change ratio (Y axis)
U(P1)V(P4)	60 revs, change ratio (Y axis)
U(P1)V(P5)	70 revs, change ratio (Y axis)
U(P1)V(P5)	80 revs, change ratio (Y axis)
U(P1)V(P6)	90 revs, stop (Y axis)
U(P1)V(P6)	100 revs
U0V0	If you are going to disable the external time base so you can move the axis independently, be sure that the axis is stopped before switching from i163 to i160. Switching the time base "on the fly", is possible; just take care that a big velocity change will not occur.
END	End of the program
	P1= the wire traversed each 10 revs of the spindle in encoder counts for the X axis P2, P3, P4, P5, P6 same for the Y axis. Just send the new variable parameters to change the program. All P variables represent the distance to move the traverse axis in encoder counts.

i28[i48] Select the Handwheel Mode Control (If i13 = 0)

Range (0 - 3), Default (3)

Modes 0 - 3 specify the external handwheel input. Modes 4 - 7 are not used. The Parabolic SMCC cannot use the other-axis encoder as handwheel input.

- 0: Standard Mode – The machine's zero position is not changed or translated. The position follows the handwheel move and the position display changes – inactive during a calculated move.
- 1: Position Offset Mode – The machine's zero position is changed and is translated as the handwheel moves. The position display does not change - active during run or stop.
- 2: 1/T Mode – The handwheel is disabled and its counters are used for special velocity decoding mode. Jumper E40 must be removed for the X axis to work in 1/T mode, and E41 must be removed for the Y axis. Also, i27[i47] must be set for proper velocity scaling (15616 for the default servo cycle time). See the description for i27[i47].
- 3: The handwheel position control is disabled but the specified handwheel signal source (external or other axis) may be used for time synchronization (see i16).

i29[i49] Set the Handwheel Encoder Control Bits (If i13 = 0)

i29 Range (0, 1, 2, 3, 16, 17, 18, 19)

i49 Range (0, 1, 2, 3, 8, 9, 10, 11)

default (3), Unit: None

This parameter controls the direction and multiplication factor of the handwheel encoder's A/B quadrature input. Also, it allows a pulse and direction input mode to be used where the A channel is used for pulse input and the B channel for direction.

i29[i49]	MULT. FACTOR	TYPE OF INPUT	DESCRIPTION
0[0]		Pulse and Direction	CW (counts up when B is high)
1[1]	x 1	Quadrature Decoding	CW (counts up when B leads A)
2[2]	x 2	Quadrature Decoding	CW (counts up when B leads A)
3[3]	x 4	Quadrature Decoding	CW (counts up when B leads A)
16[8]		Pulse and Direction	CCW (counts up when B is low)
17[9]	x 1	Quadrature Decoding	CCW (counts up when A leads B)
18[10]	x 2	Quadrature Decoding	CCW (counts up when A leads B)
19[11]	x 4	Quadrature Decoding	CCW (counts up when A leads B)

All other values are not allowed.

When using the handwheel, be sure that the handwheel inputs are enabled (see i28[i48]).

i30[i50] Set the Acceleration Feedforward Gain (If i13 = 0)

Range (0–255), Default (0), Unit: None

The acceleration feedforward gain allows the user to reduce the following error generated during accel and decel. i30[i50] generates a term which is proportional to the acceleration and adds it to its servo error output. All other servo gains should be adjusted satisfactorily first and then i30[i50] should be increased gradually until a desirable following error during accel/decel is achieved.

i31[i51] Set the Jog Feedrate (If i13 = 0)

Range (0–255), Default (200) Unit: None

This parameter sets up the speed of the jogging operation. Jog feedrate is calculated by the following equation:

$$\text{Jog Feedrate} = (i31[i51] / 256) * i06$$

Note that the jog feedrate is affected by both i06 and i31[i51]

Jog speed also can be changed by using the “%” or the “POT” command. This, however, will also affect the time values used in the program.

i32[i52] Set the Normal PWM Limit to the Control (If i13 = 0)

Range (0–255), Default (128) Unit: None

Limits the error output to the control. 0 is no output, 255 is full output or +/- 10 volts.

i33[i53] Set the Protective PWM Limit to the Control (If i13 = 0)

Range (0 - 255), Default (64), Unit: None

Sampling Rate: 43 mSec

Suggested Setup: $i32[i52] > 2 \times i33[i53]$

This parameter sets up the maximum allowable PWM limit. Once 75% of i32[i52] is reached and maintained for approximately three seconds, the SMCC goes into protective mode which uses the protective PWM limit value (i33[i53]) until the current goes down to the corresponding value. It then restores the output limit to the i32[i52] level. The I²t output and LED indicator are turned on during the protective cycle.

i34[i54] Set the Backlash Compensation (If i13 = 0)

Range (0 - 65535), Default (0), Units: Encoder Counts

This parameter allows the SMCC to compensate automatically for backlash between the motor and the load. When the direction is changed, the SMCC will add i34[i54] counts to the length of the move. For instance, if your last move was negative ending at $X = 0$, and i34 was set to 100 counts, then an X2000 move would actually cause a move of 2100 encoder counts. If the next command was X1500, there would be a move of -600 counts.

The position, as shown on the LED/LCD display or sent in response to the "p" command, does not reflect the changes caused by i34[i54]. In the example above, it would show rest positions of 0, 2000, and 1500. (A position observed when a move starts in a new direction may show the backlash. In the above example, you may see -100 and 2100 displayed for an instant.)

i34[i54] does not affect jog moves or handwheel following moves, except to note the direction of movement. If the next programmed move is in the opposite direction from the last jog or handwheel move, i34[i54] will be added to the length of the programmed move.

i35[i55] Set the Software Positive Position Limit (If i13 = 0)

Range (-1 to $2^{26}-1$),

Default (-1), Unit: Encoder Count

i35[i55] are used to prevent motion beyond the set limits. A value of -1 sets the limits to their maximum value. Caution must be exercised, since the SMCC will travel beyond the set limits by an amount equal to the distance needed for deceleration plus a maximum of five milliseconds of delay due to background servo tasks.

i36[i56] Set the Software Negative Position Limit (If i13 = 0)

Range (-1 to $2^{26}-1$),

Default (-1), Unit: Encoder Count

i36 serves the same purpose as i35. A -1 value sets the limits to the maximum (effectively no limit). i36 takes a positive value, but means a negative limit (e.g. i36 20000 sets the negative position limit to -20000).

Both of these parameters are disabled when a homing function is being performed.

i37[i57] Set the Position Range (If i13 = 0)

Range (+/- $(2^{27}-1)$).

Default (0), Unit: Encoder Count

If a negative number is specified, the range is set up for both directions up to the specified absolute value. If a positive number is specified, the range is set up for the positive direction only (1 – specified range). Set i37[i57] to zero to disable this function.

i38[i58] Set the Position and Display Scale and Format (If i13 = 0)

(n:m.d), Default (1:1.0)

Range:

n $(-2^{31} \text{ to } 2^{31} - 1)$

m $(-2^{31} \text{ to } 2^{31} - 1)$

d (0 – 7)

m/n (0 – 32)

Desired display and motion data = Encoder counts * $(m/n)^{10-d}$

The scale factor can be used to set up the display and data entry so as to represent the machine's moves in engineering units rather than encoder counts. This multiple-part parameter controls the input and output scaling of position data. Parts m and n define the conversion ratio, and d sets the order of magnitude of the conversion, plus the number of digits used to the right of the decimal point.

Example A.3

The motor has a 500 line/rev. encoder. Lead screw has a pitch of 5. It is desired to display the position in inches with 2 decimal places.

- 1) Multiply the encoder by 4, using i39[i59] value of 3 or 7.
- 2) You now have 2000 encoder counts/Rev.
- 3) The 5 pitch lead screw requires 5 revolutions for 1" of travel. $5 \times 2000 = 10000$ encoder counts/inch.
- 4) Set i38[i58] so that n=10000 (convert to inches), m=100 (show hundredths of an inch), and d=2 (decimal point two places): n can be reduced to 100 and n to 1 because it is their ratio that is important, so the command becomes i38[i58] 100:1.2
- 5) The position displayed will now show "1.00" for 1 inch of displacement.

Note that when i38[i58] is active, the input data to the SMCC must also be entered with the scaled format. An "X0.50" entry in the above example will move one half inch. Any further digits to the right of the decimal point (more than d digits) will be ignored.

The leading zero MUST be used for fractional counts for example X0.25 is okay, X.25 is not.

Important note: This parameter only affects the position values as they enter and leave the SMCC (i.e. the scaling is done as part of the transmission process). This means that this scaling factor does not affect P-variable moves [e.g. X(P1) Y(P2)].

Distances and positions specified by P-variables must always be in encoder counts, regardless of the i38[i58] scaling factors in effect.

i39[i59] Set the Position Encoder Mode Control Bits (If i13 = 0)

Range (07), Default (3)

i39[i59]	MULT. FACTOR	TYPE OF INPUT	DESCRIPTION
(0)		Pulse and Direction CW	(counts up when B is high)
(1)	x 1	Quadrature Decoding	(counts up when B leads A)
(2)	x 2	Quadrature Decoding	(counts up when B leads A)
(3)	x 4	Quadrature Decoding	(counts up when B leads A)
(4)		Pulse and Direction CCW	(counts up when B is low)
(5)	x 1	Quadrature Decoding	(counts up when A leads B)
(6)	x 2	Quadrature Decoding	(counts up when A leads B)
(7)	x 4	Quadrature Decoding	(counts up when A leads B)

As shown in the above selections, the position encoder may be selected as A/B quadrature, multiplied by 1, 2, or 4 and direction may also be selected. Note that reversing the encoder direction may also be accomplished by swapping the "A Channel" signals for "B Channel". Also, a pulse (A Channel) and direction (B Channel) (Hi = positive direction) may be specified as an encoder input (See i29[i49] for controlling the operating mode of the handwheel encoder.)

NOTE: The following definitions of i40 through i59 are only in effect if i13=1 (Brushless motor mode). If i13=0, i40 through i59 are the Y-axis equivalents of i20 through i39.

i40 Set the Encoder Counts per Electrical Revolution (If i13 = 1)

(per pole pair or cycle, for AC induction and brushless DC motors only)

Range (0 – 65535), Default (1000), Unit: Encoder Counts

The correct setup of this parameter is critical because the internal phase commutation requires this information. Note that the units are in counts per electrical revolution which is calculated by:

Counts per electrical rev. (i40) = $(2 / \text{num. of poles}) * \text{Encoder counts per mechanical rev.}$

If a 1000 count per revolution encoder is installed in a four pole motor, i40 should be 500. The default value of 1000 corresponds to a 1000 count/rev encoder for a four pole motor.

Examples:

Encoder Counts per Rev	Number of Motor Poles	Calculated value for i40
2000	4	1000
2000	2	2000
1000	4	500
1000	2	1000
500	4	250
500	2	500

NOTE: On the parabolic SMCC, if i50 is set to one, i40 must be set to the number of counts in three electrical cycles of the motor.

i41 Set the Phase Advance Scale Factor (If i13 = 1)

Range (0 – 255), Default 64, Unit: None

The commutation phase advance value is proportional to the product of shaft speed and i41. An initial adjustment is necessary to obtain the maximum torque at the desired maximum speed (adjust for minimum current). Once this adjustment at maximum speed is made, the SMCC will then automatically adjust the phase advance as a function of the velocity to optimize the current consumption.

This parameter adds a phase advance proportional to the velocity. For a speed of V in rps, the phase advance will be $i41 * V * k$. The k is different for standard and parabolic.

$k = 45/16384$ for standard (.082 degrees for 1800 RPM per increment of i41)

$k = 45/1024$ for parabolic (1.32 degrees for 1800 RPM per increment of i41)

i42 [PARAB] Induction Motor Magnetization Current (If i13 = 1)

Range (0 – 255), Default (0), Unit: None

(i42 is not used for AC induction motors on the standard SMCC. This function is controlled by i52 on the standard card.)

In the parabolic SMCC, when an induction motor control is selected by i13=1 (select brushless motor) and i45 > 0 (slip gain), then i42 is used to set the magnetization current level of the AC induction motor. The current level is dependent on the amplifier used. Normally a 5% to 10% value of maximum current is chosen; therefore, a value of i42 of 5 to 25 can be used. The higher i42 is, the greater the torque and the warmer the motor will get, even at standstill.

i43 Set the Commutation Method (If i13 = 1)

Range (129 - 131), Default (130), Unit: None

(129): 6-Step mode

(130): 3-Phase Sinusoidal mode

(131): 4-Phase Sinusoidal mode

i44 Select Number of Motor Phases (If i13 = 1)

Range (0,1), Default (1), Unit: None

0: 4-Phase motor

1: 3-Phase motor

This parameter sets the phase shift between the phase 1 and phase 3 outputs to 90 degrees for 4-phase and 120 degrees for 3 phase motors.

i45 [STD] Set the Hall Sensor Phase Offset (If i13 = 1)

Range (± 32767), Default (0), Unit: Encoder Counts

For the parabolic SMCC, parameter i54 controls this function. This parameter is used to correct mechanical misalignment between the hall effect physical position and the actual phase position.

i45 [PARAB] Set the Induction Motor Slip Gain (If i13 = 1)

Range (± 32767), Default (0), Unit: None

Refer to i51 for description.

i46 Set the Y-User Flag (If i13 = 1)

Default (0), Unit: None

i47 n/a (If i13 = 1)

Default (4096), Unit: None

i48 Set the Star - Delta Connection (If i13 = 1)

Range (0 - 1), Default (1), Unit: None

0: Brushless Motor Star (Y) connected

1: Brushless Motor Delta (Δ) connected

This parameter controls the manner in which the SMCC performs its phase finding when used with a brushless motor.

i49 n/a (If i13 = 1)

default (3), Unit: None

i50 [STD] Set the Encoder / Resolver Selection (If i13 = 1)

Range (0 , 1), Default (0), Unit: None

0: Use standard phasing initialization (assumes encoder)

1: Initializes phasing with a resolver

NOTE: On the parabolic SMCC, this function is controlled by i72.

i50 [PARAB] Enable the Triple-Cycle Motor Definition (If i13 = 1)

Range (0 , 1), Default (0), Unit: None

Setting this parameter to one permits the use of encoders and resolvers whose number of counts per revolution is not divisible by three in cases where the brushless or induction motor has 3 or 6 cycles (6 or 12 poles). On the standard SMCC, or with i50=0, there must be an integer number of counts per electrical cycle of the motor (i40 is set to this value). When i50 is set to one, i40 must be set to the number of counts in three electrical cycles of the motor, so the number of counts in one cycle does not have to be an integer. This is particularly useful for using absolute encoders or resolvers, whose counts per revolution are typically powers of two, with 6-pole or 12-pole brushless motors.

i51 [STD] Set the Induction Motor Slip Gain (If i13 = 1)

Range (0 - 255), Default (0) Unit: None

NOTE: This function is controlled by i45 in the parabolic SMCC (where it has a range of 0 - 65535). i51 is not used in the parabolic SMCC.

If this value is not zero, then i52 controls the magnetization PWM and induction motor control is enabled (see i52). When i51 is in use (non-zero), it controls the slip frequency fluctuations needed to control the AC induction motor's torque.

This term is used for induction motors only and is used to produce a slip frequency that is proportional to the torque required. The range and scale is different for standard and parabolic. The range for standard is 0 to 255 and 0 to 65535 for parabolic. The slip frequency is $T \cdot i51[i45] \cdot k$ lines per second where T is the torque current in PWM units (255=full scale) and k is a constant that depends on the SMCC type. $k=1024/256$ for standard (1024 is the servo update rate) and $k=5859.375/65536$ for parabolic (5859.375 is the phasing update rate).

i52 [PARAB] Set the DC Brushless Motor Phase Finding Time (If i13 = 1)

Range (0 - 255), Default (20) Unit: 43 milliseconds

[STD] Set the AC Induction Motor Magnetization Current (If i13 = 1)

Range (0 - 255), Default (0) Unit: None

(On the parabolic SMCC, this function is controlled by i42.)

For a DC brushless motor (when i13=1 and i51 (STD) i45 (PARAB) is set to zero), i52 controls the length of time allocated to the automatic phase finding that must take place upon power turn-on or reset. Generally, phase finding requires from 1/2 to 2 seconds.

For an AC induction motor (when i51 (STD) or i45 (PARAB) is not set to zero), phase finding is not necessary, and i52 controls the quiescent magnetization current into the motor, i52 0 set 0 current and i52 255 sets maximum current available from the amplifier. Caution must be used not to set this value too high since it will tend to overheat the motor even while it is at rest. A typical setting is i52 5 to 20.

NOTE: On the standard SMCC, if i51 is greater than zero (this selects an induction motor since brushless DC motors have no slip gain), then i52 = magn. Current. If i51 is zero, then i52 is the phase finding time.

i53 Set the Phasing Current Value (If i13 = 1)

Range (0 – 255), Default (20), Unit: None

Sets the amount of current delivered to the motor (brushless) when the phases are being located upon power turn-on or reset. This motion causes the motor to move approximately +/- 60 degrees of shaft rotation. Note that if hall effects are installed, the phase finding motion is reduced to looking for the first hall effect edge transition and that the direction of motion can then be controlled and occurs at full torque.

i54 [PARAB] Set the Hall Effect or Resolver Offset (If i13 = 1)

Range (+/- 32767), Default (0), Units: Counts

(See i45 for an explanation of the phase offset operation for hall effect, phasing, or resolver alignment.)

NOTE: Please refer to the installation and startup procedure for starting up brushless and AC induction motors. Parameters i15 and i40 to i54 all pertain to these types of motors.

i55 n/a (If i13 = 1)

Default (-1), Unit: None

i56 n/a (If i13 = 1)

Default (-1), Unit: None

i57 n/a (If i13 = 1)

Default (0), Unit: None

i58 n/a (If i13 = 1)

Default (1:1.0), Unit: None

i59 n/a (If i13 = 1)

Default (3), Unit: None

i60 [STD] PLC Enable

Range (0 , 1), Default (0), Unit: None

This parameter is used to activate or deactivate the PLC program that is stored in the SMCC. If i60=1, the PLC program will be active at all times, regardless of whether a regular program is executing or not. The only exception to this is during the loading of a PLC program, when the SMCC automatically disables the PLC (from the time it receives an “IF” command until it receives an “END” command).

i60[i61] [PARAB] Deadband

Range (0 - 255), Default (0), Unit: None

Specifies a +/- position range from the desired position where the SMCC does not close the servo loop. This is useful in systems that have a lot of backlash between the motor and the load.

Note: Parameters i61 and greater are for the parabolic version only.

i62 PLC Enable

Range (0, 1), Default (0), Unit: None

This parameter is used to activate or deactivate the PLC program stored in the SMCC. Set it to 1 and the PLC program will be active at all times, regardless of whether a regular program is executing or not. The only exception to this is during the loading of a PLC program.

i63 Mode Control for DMA Communication

Range (0 - 1), Default (0), Unit: None

i63 determines what data the parabolic SMCC expects when it is in DMA mode. When i63 = 0, it expects binary position and velocity data for each axis (to define a parabolic move segment). When i63 = 1, it expects only binary position data (to define a linear move segment).

i64 - i77

Reserved, default value 0.

i78[i79] Range Mode Control (Shortest Distance or Absolute)

Range (0 - 1), Default (0)

See parameter i37[i57] which controls the SMCC's position range.

i78[i79] = 0, selects the shortest distance to travel from one position to the next.

i78[i79] = 1, selects a direction controlled move within the range

i80 Backlash Take-up (Parabolic version 2.78 and newer)

Range (0 - 15), Default (0)

When i80 = 0, the backlash function is the same as on older versions of the SMCC, it is only active on programmed moves, not jog or handwheel moves. When i80 > 0, backlash compensation is also active on jog or handwheel moves, and the value of i80 determines the number of counts per background cycle (approximately 4 mSec.) at which the backlash is taken up on direction reversal.

i81 Circle Error Tolerance (Parabolic version 2.78 and newer)

Range (0 - 15), Default (0), units: encoder counts

This parameter allows a circular move of a distance slightly greater than twice the radius to be executed without error (this situation usually occurs due to round off errors when specifying a half-circle).

If the specified move distance is greater than twice the radius by a number of encoder counts less than i81, the radius is increased slightly so that a half-circle move is made.

If the error is greater than i81, the program will stop and an error will be generated.

Appendix B

SMCC Status Codes

B.1 SMCC ON-BOARD LED DISPLAY CODES

Following Error:

- .(Left Decimal) = In Position indicator (See Appendix A, i01)
- .(Right Decimal) = Excess Following Error Indicator (i10)
(Indication only, See Appendix A, i10)

SMCC Status

- 0 = Okay (SMCC is Operating Normally)
- 1 = 1 Reserved
- 2 = Following Error Failure (i00)
- 3 = Not enough time to calculate the move
- 4 = The circle radius was too small for specified radius.
- 5 = The "Go Until" command did not receive a trigger
- 6 = The "Go Until" command did not receive a trigger
- 7 = Communications Error (Invalid Character)
- 8 = Control Failure, (without Fault Data Bits)
- code 9, A, B, C, D, E, F Reserved

B.2 SMCC STATUS BYTES

The SMCC will send the Status bytes upon the reception of a "?" and a carriage return ("?" is status query command).

BIT	Byte 1 (feb5)
0	Home Complete (Y)
1	Home Complete (X)
2	Continuous Move
3	Block Request (MDI Mode)
4	Auto Divide
5	Feed Hold
6	Running (any move)
7	Timer Active

BIT	Byte 2 (feb6)
0	Time Mode (not feedrate)
1	Single Step
2	Buffer Full
3	Parabolic Mode
4	Limit Switch
5	Not Ready (X)
6	Not Ready (Y)
7	Not Ready For Move

BIT	Byte 3 (feb7)
0	X Jog Positive
1	X Jog Negative
2	X Limit Pos. (soft or hard)
3	X Limit Neg. (soft or hard)
4	X Foll. Err. (i10 exceeded)
5	X I ² T Limit (low true)
6	X Not in Position
7	X Zero Velocity (commanded)

BIT	Byte 4 (feb8)
0	Y Jog Positive
1	Y Jog Negative
2	Y Limit Pos. (soft or hard)
3	Y Limit Neg. (soft or hard)
4	Y Foll. Err. (i10 exceeded)
5	Y I ² T Limit (low true)
6	Y Not in Position
7	Y Zero Velocity (commanded)

BIT	Byte 5 (feb9)
0	Edit Mode
1	Insert Mode
2	Extended Read
3	Circle Mode
4	Wait (X)
5	Wait (Y)
6	Backlash (X)
7	Backlash (Y)

BIT	Byte 6 (feba)
0	Auto Divide Req
1	Extended Move Flag
2	Extended Move Flag
3	Home Has Been Done Flag
4	CCW Circle
5	Circle > 180
6	Position Frozen (after ^V)
7	Sub. Buffer Open

BIT	Byte 7 (febb)
0	Servo Pointer
1	Servo Pointer
2	Servo Pointer
3	Dwell Request
4	Home in Progress until capture
5	Dwell in Progress
6	Continuous Move Request
7	Memory Request

BIT	Byte 8 (febc)
0	Servo Pointer
1	Servo Pointer
2	Servo Pointer
3	Servo Pointer
4	Servo Pointer
5	Step Request
6	Servo Pointer
7	Servo Pointer

Appendix C

On-Board DIP Switch

C.1 CARD ADDRESS SELECTION SWITCH

CARD ADDRESS SELECTION SWITCH				
SW1	SW2	SW3	SW4	CARD ADDRESS
0	0	0	0	A0
1	0	0	0	A1
0	1	0	0	A2
1	1	0	0	A3
0	0	1	0	A4
1	0	1	0	A5
0	1	1	0	A6
1	1	1	0	A7
0	0	0	1	A8
1	0	0	1	A9
0	1	0	1	Aa
1	1	0	1	Ab
0	0	1	1	Ac
1	0	1	1	Ad
0	1	1	1	Ae
1	1	1	1	Af

C.2 EAROM ENABLE SWITCH

DIP switch No. 5 is used to Enable (0) or Disable (1) saving programs and parameters into EAROM memory

C.3 COMMUNICATION MODE AND BAUD RATE SELECTION SWITCH

Communication Mode and Baud Rate Selection Switch			
SW6	SW7	SW8	COMMUNICATION MODE
0	0	0	Serial 9600 Baud
1	0	0	Serial 4800 Baud
0	1	0	Serial 2400 Baud
1	1	0	Serial 1200 Baud
0	0	1	Parallel Communication
1	0	1	Serial 300 Baud
0	1	1	Serial 38400 Baud
1	1	1	Serial 19200 Baud

Appendix D

SMCC Memory and I/O Map

The Parabolic SMCC's memory and register map are provided for the purpose of allowing users to have direct access to the many variables available in the SMCC so that more sophisticated programs, especially in PLC, may be written to accomplish more useful and difficult tasks. A brief summary is provided here to point out some of the commonly used registers:

Frequently Used Memory

<u>Address</u>		<u>Description</u>
2034	5 bytes	Version number address
FF01	Bit 0	STRT, Panel Interface, (RUN or Start Button)
FF01	Bit 1	STEP, Panel Interface, (STEP Button)
FF01	Bit 4	I ² T/
FF01	Bit 5	MA01, Machine Output #1
FF01	Bit 6	FERR
FF01	Bit 7	IPOS/

SERVO 2 = 1 (See Note)

FF3A	Bit 4	YMIL, Panel Interface, (Y axis negative limit)
FF3A	Bit 5	YPLL, Panel Interface, (Y axis positive limit)
FF3A	Bit 6	YJPL, Panel Interface, (Y axis Jog positive)
FF3A	Bit 7	YJMI, Panel Interface, (Y axis Jog negative)

SERVO 1 = 1 (See Note)

FF3A	Bit 4	XMIL, Panel Interface, (X axis negative limit)
FF3A	Bit 5	XPLL, Panel Interface, (X axis positive limit)
FF3A	Bit 6	XJPL, Panel Interface, (X axis Jog positive)
FF3A	Bit 7	XJMI, Panel Interface, (X axis Jog negative)

Note:

SERVO 1 – when SYNC/ line is Low.

SERVO 2 – when SYNC/ line is High.

EQU= XJMI ANDed XJPL, Return to PreJog Position.

HOME= YJMI ANDed YJPL, Start Home cycle

HOLD= Any Pair of XPLL, XMIL, YPLL, YMIL ANDed

QUIT= RUN AND STEP

FFB4	Bit 6	MA02, Machine Output #2.
FFB4	Bit 7	MA03, Machine Output #3.
FFB5	Bit 6	MA04, Machine Output #4.
FFB5	Bit 7	MA05, Machine Output #5.
FFB7	Bit 0	MAI1, Machine Input #1.
FFB7	Bit 1	MAI2, Machine Input #2.
FFB7	Bit 2	MAI3, Machine Input #3.
FFB7	Bit 3	MAI4, Machine Input #4.
FFB7	Bit 4	XHFL/, X axis Home Input.

FFB7	Bit 5	XUFL/, X axis (Alternate Home) Input.
FFB7	Bit 6	YHFL/, Y axis Home Input.
FFB7	Bit 7	YUFL/, Y axis (Alternate Home) Input.
FFB8	1 byte	Parallel Port Data Address, DAT0–7 correspond to Bits 0–7
FFB9	1 byte	Parallel Port Select Line Address, SEL0–7 correspond to Bits 0–7.
FFBC	Bit 0	MAI5, Machine Input #5.
FFBC	Bit 1	MAI6, Machine Input #6.
FFBC	Bit 2	MAI7, Machine Input #7.

BIT MAP FOR MAPPING MACHINE INPUT/OUTPUT

The SMCC automatically sets the following definitions:

Address Definition	Machine Outputs	Machine Inputs
C01 = FF01	M01 is M01 = C01.5	MI1 is M11 = C04.0
C02 = FFB4	M02 is M02 = C02.6	MI2 is M12 = C04.1
C03 = FFB5	M03 is M03 = C02.7	MI3 is M13 = C04.2
C04 = FFB7	M04 is M04 = C03.6	MI4 is M14 = C04.3
C05 = FFBC	M05 is M05 = C03.7	MI5 is M15 = C05.0
		MI6 is M16 = C05.1
		MI7 is M17 = C05.2

MINI INTERNAL RAM PARABOLIC

```

                ORG 0FE00H ;INTERNAL RAM
FE00          RBUFF:      DS 32
              FE00 TBUF    EQU RBUFF
                        ;START OF DIRECT ACCESS RAM
FE20          PTM1:       DS 2          ;PREVIOUS ENCODER EDGE TIME X-AXIS
FE22          PTM2:       DS 2          ;PREVIOUS ENCODER EDGE TIME Y-AXIS
              FE22 PPHASE  EQU TPTM2    ;PREVIOUS PHASE ENCODER READING
FE24          PDPOS1:     DS 2          ;PREVIOUS DESIRED POSITION X-AXIS
FE26          PDPOS2:     DS 2          ;PREVIOUS DESIRED POSITION Y-AXIS
              FE26 PHASE   EQU PDPOS2   ;PRESENT PHASE POSITION
FE28          DACR1:      DS 2          ;RESIDUAL DAC OUTPUT X-AXIS
FE2A          DACR2:      DS 2          ;RESIDUAL DAC OUTPUT Y-AXIS
              FE2A DAC     EQU DACR2    ;DAC OUTPUT FOR PHASED MOTORS
FE2C          PRES1:      DS 4          ;POSITION RESIDUAL X-AXIS
FE30          PRES2:      DS 4          ;POSITION RESIDUAL Y-AXIS
              FE30 SLIP    EQU PRES2    ;SLIP FREQUENCY
FE34          VRES1:      DS 2          ;VEL RESIDUAL X-AXIS
FE36          VRES2:      DS 2          ;VEL RESIDUAL Y-AXIS
              FE36 PHASAD  EQU VRES2    ;PHASE ADVANCE
FE38          ARES1:      DS 2          ;ACCEL RESIDUAL X-AXIS
FE3A          ARES2:      DS 2          ;ACCEL RESIDUAL Y-AXIS
              FE3A PHASOA  EQU ARES2    ;
FE3C          HWRES1:     DS 1          ;HANDWHEEL RESIDUAL X-AXIS
FE3D          HWRES2:     DS 1          ;HANDWHEEL RESIDUAL Y-AXIS
FE3E          PHW1:       DS 2          ;PREVIOUS HANDWHEEL X-AXIS
FE40          PENC1:      DS 2          ;PREVIOUS ENCODER X-AXIS
FE42          IPOS1:      DS 4          ;ERROR INTEGRAL X-AXIS
FE46          PDVEL1:     DS 2          ;PREVIOUS DESIRED VEL X-AXIS
FE48          APOS1:      DS 4          ;ABSOLUTE POSITION X-AXIS
FE4C          APOS2:      DS 4          ;ABSOLUTE POSITION Y-AXIS
FE50          EPOS1:      DS 4          ;ERROR POSITION X-AXIS
FE54          EPOS2:      DS 4          ;ERROR POSITION Y-AXIS
              FE54 PHRES   EQU EPOS2    ;RESIDUAL PHASE POSITION

FE58          DPOS1:      DS 4          ;DESIRED POSITION X-AXIS
FE5C          DPOS2:      DS 4          ;DESIRED POSITION Y-AXIS
FE60          DVEL1:      DS 4          ;DESIRED VEL X-AXIS
FE64          DVEL2:      DS 4          ;DESIRED VEL Y-AXIS
FE68          ACCEL1:     DS 6          ;DESIRED ACCEL X-AXIS
FE6E          ACCEL2:     DS 6          ;DESIRED ACCEL Y-AXIS
FE74          JCCCEL1:    DS 8          ;DESIRED ACCEL DOT X-AXIS

```

FE7C	JCCCEL2:	DS	8	;DESIRED ACCEL DOT Y-AXIS
FE84	NMVTIM:	DS	4	;NEW MOVE TIME
FE88	MOVTIM:	DS	4	;MOVE TIME
FE8C	FATD0:	DS	2	;FILTERED ATD #0 INPUT
FE8E	FATD1:	DS	2	;FILTERED ATD #1 INPUT
FE90	ATD2:	DS	1	;RAW ATD #2 INPUT
FE91	ATD3:	DS	1	;RAW ATD #3 INPUT
FE92	DFDPOT:	DS	2	;DESIRED FEED POT
FE94	FDSLEW:	DS	2	;FEED POT SLEW RATE
FE96	AFDPOT:	DS	2	;FEED POT USED FOR INTEGRATION
FE98	HFD POT:	DS	2	;FEED HOLD FEED POT
FF8E	I05	EQU	0FF8EH	;SERVO TIME
FE9A	I07:	DS	2	;DWELL TIME BASE
FE9C	I20:	DS	2	;AXIS #1 PROPORTIONAL GAIN
FE9E	I27:	DS	2	;AXIS #1 HANDWHEEL SCALE FACTOR
FEA0	I40:	DS	2	;AXIS #2 PROPORTIONAL GAIN
FEA2	I45:	DS	2	;AXIS #2 HOME OFFSET/SLIP GAIN
FEA4	I47:	DS	2	;AXIS #2 HANDWHEEL SCALE FACTOR
FEA6	I21:	DS	1	;AXIS #1 FEEDBACK GAIN
FEA7	I22:	DS	1	;AXIS #1 FEEDFORWARD
FEA8	I23:	DS	1	;AXIS #1 INTEGRAL GAIN
FEA9	I32:	DS	1	;PWM #1 LIMIT
FEAA	I33:	DS	1	;I ² T PWM #1 LIMIT
FEAB	I41:	DS	1	;AXIS #2 FEEDBACK GAIN
FEAC	I42:	DS	1	;AXIS #2 FEEDFORWARD
FEAD	I43:	DS	1	;AXIS #2 INTEGRAL GAIN
FEAE	I52:	DS	1	;PWM #2 LIMIT
FEAF	I53:	DS	1	;I ² T PWM #2 LIMIT
FEB0	IBITS1:	DS	1	
0580	I02	EQU	IBITS1.0	;MDI MODE
0581	I03	EQU	IBITS1.1	;HANDSHAKE MODE (2 BITS)
0582	I03B1	EQU	IBITS1.2	
0583	I11	EQU	IBITS1.3	;FEED HOLD SLEW
0584	I13	EQU	IBITS1.4	;BRUSHLESS MOTOR ENABLE
0585	I15	EQU	IBITS1.5	;Y-AXIS DISABLE/(TRAP/LOOKUP;PHASING)
0586	I14	EQU	IBITS1.6	;A TO D MODE (2 BITS)
0587	I14B1	EQU	IBITS1.7	

FEB1	IBITS2:	DS 1	
0588	I24	EQU IBITS2.0	;HOME DIRECTION #1
0589	I28	EQU IBITS2.1	;HANDWHEEL MODE #1 (2 BITS)
058A	I28B1	EQU IBITS2.2	
058B	I09	EQU IBITS2.3	;INTEGRATOR MODE
058C	I44	EQU IBITS2.4	;HOME DIRECTION #2
058D	I48	EQU IBITS2.5	;HANDWHEEL MODE #2 (2 BITS)
058E	I48B1	EQU IBITS2.6	
058F	PBFULL	EQU IBITS2.7	;PROGRAM BUFFER CLOSED FLAG
FEB2	IBITS3:	DS 1	
0590	I18	EQU IBITS3.0	;DISPLAY MODE (2 BITS)
0591	I18B1	EQU IBITS3.1	
0592	I16	EQU IBITS3.2	;FEEDPOT HANDWHEEL MODE
0593	I16B1	EQU IBITS3.3	
0594	I19	EQU IBITS3.4	;HIGHEST CARD IN ADDRESSED;(4 BITS)
FEB3	I30:	DS 1	;AXIS #1 DES. VEL FEEDBACK GAIN
FEB4	I50:	DS 1	;AXIS #2 DES. VEL FEEDBACK GAIN
FEB5	BITS:	DS 2	
05A8	HOME2	EQU BITS.0	;HOME AXIS #2 FLAG
05A9	HOME1	EQU BITS.1	;HOME AXIS #1 FLAG
05AA	CMOVE	EQU BITS.2	;CONTINUOUS MOVE FLAG
05AB	BLKREQ	EQU BITS.3	;BLOCK REQUEST FLAG
05AC	ADIV	EQU BITS.4	;AUTO DIVIDE MOVE FLAG
05AD	FHOLD	EQU BITS.5	;FEED HOLD FLAG
05AE	RUN	EQU BITS.6	;RUN FLAG
05AF	TIMER	EQU BITS.7	;TIMER ACTIVE FLAG
05B0	TMODE	EQU BITS+1.0	;TIME MODE FLAG
05B1	STEP	EQU BITS+1.1	;SINGLE STEP FLAG
05B2	BFULL	EQU BITS+1.2	;BUFFER FULL FLAG
05B3	FRATE	EQU BITS+1.3	;FEEDRATE MOVE FLAG
05B4	LIMIT	EQU BITS+1.4	;LIMIT SWITCH FLAG
05B5	NTRDY1	EQU BITS+1.5	;NOT READY AXIS #1 FLAG
05B6	NTRDY2	EQU BITS+1.6	;NOT READY AXIS #2 FLAG
05B7	NOTRDY	EQU BITS+1.7	;NOT READY FLAG
FEB7	SBITS1:	DS 1	
05B8	JOGP1	EQU SBITS1.0	;X-AXIS POSITIVE JOG
05B9	JOGM1	EQU SBITS1.1	;X-AXIS NEGATIVE JOG
05BA	LMTP1	EQU SBITS1.2	;X-AXIS POSITIVE LIMIT
05BB	LMTM1	EQU SBITS1.3	;X-AXIS NEGATIVE LIMIT
05BC	FE1	EQU SBITS1.4	;X-AXIS FOLLOWING ERROR WARNING

05BD	I ² T1	EQU SBITS1.5	;X-AXIS CURRENT LIMIT
05BE	NIPOS1	EQU SBITS1.6	;X-AXIS NOT IN POSITION
05BF	RVZ1	EQU SBITS1.7	;X-AXIS RUN VEL ZERO
FEB8	SBITS2:	DS 1	
05C0	JOGP2	EQU SBITS2.0	;Y-AXIS POSITIVE JOG
05C1	JOGM2	EQU SBITS2.1	;Y-AXIS NEGATIVE JOG
05C2	LMT P2	EQU SBITS2.2	;Y-AXIS POSITIVE LIMIT
05C3	LMTM2	EQU SBITS2.3	;Y-AXIS NEGATIVE LIMIT
05C4	FE2	EQU SBITS2.4	;Y-AXIS FOLLOWING ERROR WARNING
05C5	I ² T2	EQU SBITS2.5	;Y-AXIS CURRENT LIMIT
05C6	NIPOS2	EQU SBITS2.6	;Y-AXIS NOT IN POSITION
05C7	RVZ2	EQU SBITS2.7	;Y-AXIS RUN VEL ZERO
FEB9	XBITS:	DS 2	
05C8	EDIT	EQU XBITS.0	;EDIT MODE
05C9	INSERT	EQU XBITS.1	;EDIT INSERT MODE
05CA	EXREAD	EQU XBITS.2	;EXTENDED READ IN PROGRESS
05CB	CIRCLE	EQU XBITS.3	;CIRCLE MODE
05CC	WAIT1	EQU XBITS.4	;X-AXIS WAIT MOVE
05CD	WAIT2	EQU XBITS.5	;Y-AXIS WAIT MOVE
05CE	BACK1	EQU XBITS.6	;X-AXIS BACKLASH COMP
05CF	BACK2	EQU XBITS.7	;Y-AXIS BACKLASH COMP
05D0	ADREQ	EQU XBITS+1.0	;AUTO DIVIDE REQUEST BIT 0
05D1	EXMOVE	EQU XBITS+1.1	;EXTENDED MOVE FLAG (2 BITS)
05D2	EXMOV I	EQU XBITS+1.2	
05D3	HMDONE	EQU XBITS+1.3	;HOME DONE FLAG
05D4	CIRDIR	EQU XBITS+1.4	;CIRCLE DIRECTION FLAG
05D5	CIR180	EQU XBITS+1.5	;CIRCLE GREATER THAN 180 FLAG
05D6	FREEZE	EQU XBITS+1.6	;POSITION FREEZE FLAG
05D7	SBOPEN	EQU XBITS+1.7	;SUBROUTINE BUFFER OPEN FLAG
FEBB	SFLAG:	DS 1	
05DB	DWEREQ	EQU SFLAG.3	;DWELL REQUESTED
05DC	HOME	EQU SFLAG.4	;HOME IN PROGRESS
05DD	DWELL	EQU SFLAG.5	;DWELL IN PROGRESS
05DE	CMVREQ	EQU SFLAG.6	;CONTINUOUS MOVE REQUESTED
05DF	MFMRFO	EQU SFLAG.7	;MEMORY MODIFY REQUESTED
FEBE	BUFPTR:	DS 1	
05E5	STPREQ	EQU BUFPTR.5	;STOP REQUESTED
FEBD	DBAND1:	DS 1	
05ED	BDATA	EQU DBAND1.5	;BUFFER DATA FLAG

05EE	DMAIO	EQU DBAND1.6	;DMA IO IN PROGRESS
05EF	DMAOUT	EQU DBAND1.7	;DMA OUTPUT MODE
FEBE	DBAND2.	DS 1	
05F5	PLCFLG	EQU DBAND2.5	;PLC BUFFER OPEN FLAG
05F6	RESULT	EQU DBAND2.6	;PLC RESULT DECODE FLAG
05F7	PLCCMD	EQU DBAND2.7	;PLC COMMAND FLAG
FEBF	P01BUF:	DS 1	
05FC	UMOVIC	EQU P01BUF.4	;MOVE UNTIL INTERRUPT ENABLE
05FD	TORQUE	EQU P01BUF.5	;TORQUE MOVE IN PROGRESS BIT

ORG 0FEC0H

FEC0	MR0:	DS 1	;REGISTER BANK 3
FEC1	MR1:	DS 1	
FEC2	MR2:	DS 1	
FEC3	MR3:	DS 1	
FEC4	MR4:	DS 1	
FEC5	MR5:	DS 1	
FEC6	MR6:	DS 1	
FEC7	MR7:	DS 1	
FEC8	MR8:	DS 1	
FEC9	MR9:	DS 1	
FECA	MR10:	DS 1	
FECB	MR11:	DS 1	
FECC	MR12:	DS 1	
FECD	MR13:	DS 1	
FECE	MR14:	DS 1	
FECF	MR15:	DS 1	
FEC0	MRP0	EQU MR0	
FEC2	MRP1	EQU MR2	
FEC4	MRP2	EQU MR4	
FEC6	MRP3	EQU MR6	
FEC8	MRP4	EQU MR8	
FECA	MRP5	EQU MR10	
FECC	MRP6	EQU MR12	
FECE	MRP7	EQU MR14	
FED0	SR0:	DS 1	;REGISTER BANK 2
FED1	SR1:	DS 1	
FED2	SR2:	DS 1	
FED3	SR3:	DS 1	
FED4	SR4:	DS 1	
FED5	SR5:	DS 1	
FED6	SR6:	DS 1	
FED7	SR7:	DS 1	
FED8	SR8:	DS 1	

FED9	SR9:	DS 1	
FEDA	SR10:	DS 1	
FEDB	SR11:	DS 1	
FEDC	SR12:	DS 1	
FEDD	SR13:	DS 1	
FEDC	SR14:	DS 1	
FEDF	SR15:	DS 1	
FED0	SRP0	EQU SR0	
FED2	SRP1	EQU SR2	
FED4	SRP2	EQU SR4	
FED6	SRP3	EQU SR6	
FED8	SRP4	EQU SR8	
FEDA	SRP5	EQU SR10	
FEDC	SRP6	EQU SR12	
FEDE	SRP7	EQU SR14	
FEE0	IR0:	DS 1	;REGISTER BANK 1
FEE1	IR1:	DS 1	
FEE2	IR2:	DS 1	
FEE3	IR3:	DS 1	
FEE4	IR4:	DS 1	
FEE5	IR5:	DS 1	
FEE6	IR6:	DS 1	
FEE7	IR7:	DS 1	
FEE8	IR8:	DS 1	
FEE9	IR9:	DS 1	
FEEA	IR10:	DS 1	
FEEB	IR11:	DS 1	
FEEC	IR12:	DS 1	
FEED	IR13:	DS 1	
EEEE	IR14:	DS 1	
FEEF	IR15:	DS 1	
FEE0	IRP0	EQU IR0	
FEE2	IRP1	FQU IR2	
FEE4	IRP2	EQU IR4	
FEE6	IRP3	EQU IR6	
FEE8	IRP4	EQU IR8	
FEEA	IRP5	EQU IR10	
FFFC	IRP6	FQU IR12	
EEEE	IRP7	EQU IR14	
FEE0	IOBITS	EQU IR0	
0701	AADRSD	EQU IOBITS.1	;ALL CARDS ADDRESSED
0702	ADRSED	EQU IOBITS.2	;THIS CARD ADDRESSED
0703	RS232	EQU IOBITS.3	;RS232 MODE
FEE2	ADDRES	EQU IR2	;CARD ADDRESS

FEE3	BLNGTH	EQU IR3	;INPUT BUFFER LENGTH
FEE4	STIME	EQU IR4	;SERVO TIMER
FEE5	ACC	EQU IR5	
FEE6	PHW2	EQU IRP3	;PREVIOUS HANDWHEEL Y-AXIS
FEE8	PENC2	EQU IRP4	;PREVIOUS ENCODER Y-AXIS
FEEA	PENC4	EQU IRP5	;PREVIOUS AV.ENCODER Y-AXIS
FEF0	PR0:	DS 1	;REGISTER BANK 0
FEF1	PR1:	DS 1	
FEF2	PR2:	DS 1	
FEF3	PR3:	DS 1	
FEF4	PR4:	DS 1	
FEF5	PR5:	DS 1	
FEF6	PR6:	DS 1	
FEF7	PR7:	DS 1	
FEF8	PR8:	DS 1	
FEF9	PR9:	DS 1	
FEFA	PR10:	DS 1	
FEFB	PR11:	DS 1	
FEFC	PR12:	DS 1	
FEFD	PR13:	DS 1	
FEFE	PR14:	DS 1	
FEFF	PR15:	DS 1	
FEF0	PRP0	EQU PR0	
FEF2	PRP1	EQU PR2	
FEF4	PRP2	EQU PR4	
FEF6	PRP3	EQU PR6	
FEF8	PRP4	EQU PR8	
FEFA	PRP5	EQU PR10	
FEFC	PRP6	EQU PR12	
FEFE	PRP7	EQU PR14	
FEF4	IPOS2	EQU PRP2	;ERROR INTEGRAL Y-AXIS
FEF8	PDVEL2	EQU PRP4	;PREV. DESIRED VEL Y-AXIS
FEFA	FDPOT	EQU PRP5	;PRESENT FEED POT

EXTERNAL RAM PARABOLIC

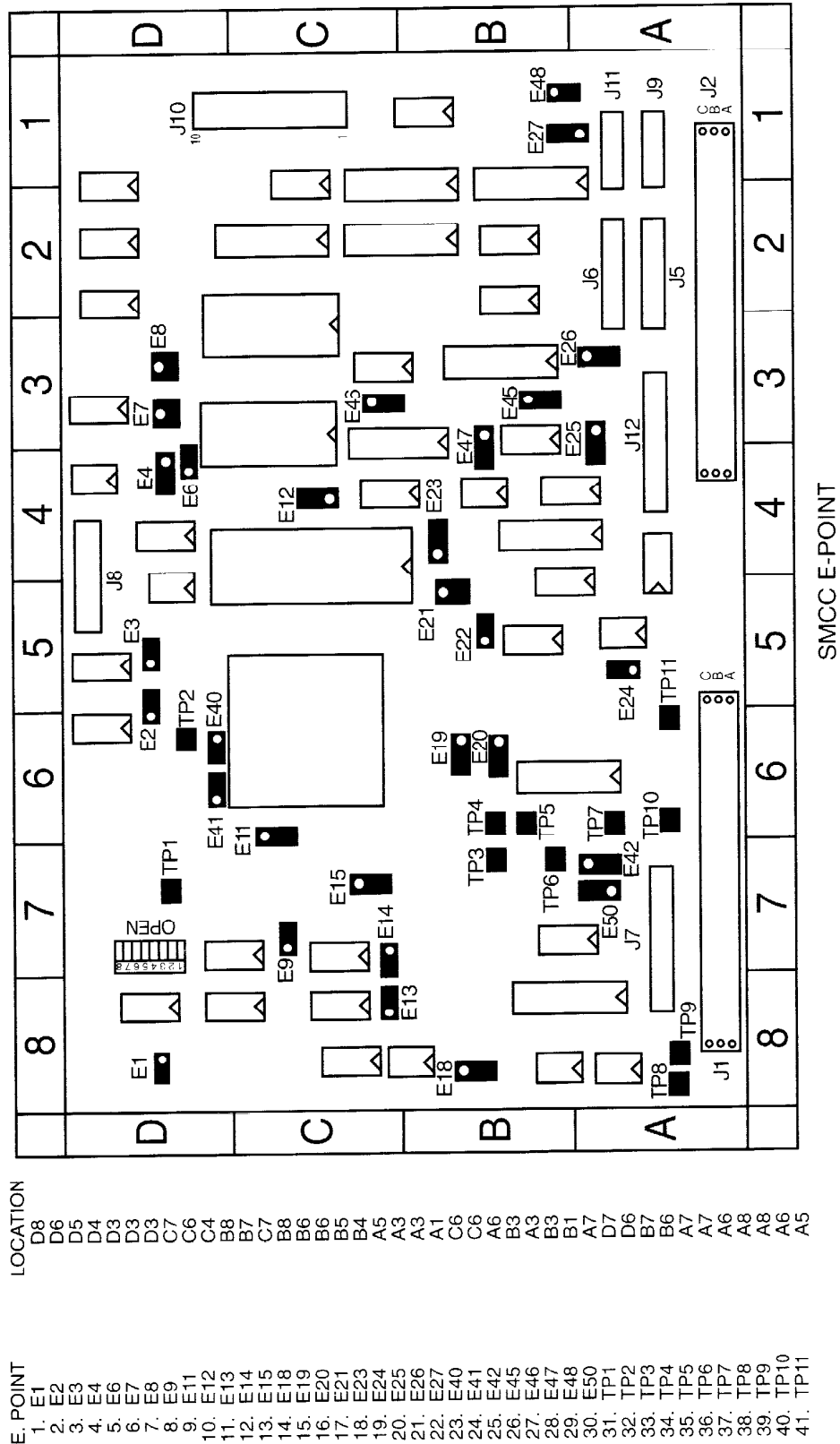
ORG 0A000H

A000	ADPOS1:	DS 4	;NEXT DESIRED POSITION X-AXIS
A004	ADPOS2:	DS 4	;NEXT DESIRED POSITION Y-AXIS
A008	ADVEL1:	DS 4	;NEXT DESIRED VEL X-AXIS
A00C	ADVEL2:	DS 4	;NEXT DESIRED VEL Y-AXIS
A010	NACEL1:	DS 6	;NEXT DESIRED ACCEL X-AXIS
A016	NACEL2:	DS 6	;NEXT DESIRED ACCEL Y-AXIS
A01C	NJCEL1:	DS 8	;NEXT DESIRED ACCEL DOT X-AXIS
A024	NJCEL2:	DS 8	;NEXT DESIRED ACCEL DOT Y-AXIS
A02C	NMVTM1:	DS 4	;NEXT DESIRED MOVE TIME
A030		DS 144	
A0C0	NDPOS1:	DS 4	;FINAL DESIRED POSITION X-AXIS
A0C4	NDPOS2:	DS 4	;FINAL DESIRED POSITION Y-AXIS
A0C8	NDVEL1:	DS 4	;FINAL DESIRED VEL X-AXIS
A0CC	NDVEL2:	DS 4	;FINAL DESIRED VEL Y-AXIS
A0D0	CCOUNT:	DS 2	;CONTINUOUS MOVE COUNT
A0D2	CDPOS1:	DS 6	;CONT.MOVE DESIRED POS. X-AXIS
A0D8	DDPOS1:	DS 6	;CONT.MOVE DELTA POS. X-AXIS
A0DE	CDPOS2:	DS 6	;CONT.MOVE DESIRED POS.Y-AXIS
A0E4	DDPOS2:	DS 6	;CONT.MOVE DELTA POS.Y-AXIS
A0EA	FDRATE:	DS 4	;FEEDRATE/TIME
A0EA	TIME	EQU FDRATE	
A0EE	TEMP:	DS 16	
A0FE	ATIME:	DS 2	;ACCEL TIME
A100	STABLE:	DS 128	;SYMBOL TABLE L00L63
A180	CTABLE:	DS 128	;SYMBOL TABLE C00C63
A200	MTABLE:	DS 64	;SYMBOL TABLE M00M63
A240	PTABLE:	DS 60	;PARAMETER TABLE P01P15
A27C	STEND:	DS 2	;STORAGE END POINTER
A27E	LBPTR:	DS 2	;SUBROUTINE LIBRARY POINTER
A280	BGPTR:	DS 2	;PLC POINTER
A282	STACK:	DS 124	;SERVO STACK
A2FE	STKEND:		
A2FE	SSTPTR:	DS 1	;SERVO STACK POINTERS
A2FF	STKPTR:	DS 1	
A300	XRBUFF:	DS 65	;COM BUFFERS
A300	XTBUFF	EQU XRBUFF	
A341	DISTIM:	DS 2	;DISPLAY TIMFR
A343	DISCNT:	DS 1	;DISPLAY COUNTER

A344	BDPTR:	DS	9	
A34D	BPOS1:	DS	4	;BIAS POSITION X-AXIS
A351	BPOS2:	DS	4	;BIAS POSITION Y-AXIS
A355	HPOS1:	DS	4	;HOME POSITION X-AXIS
A359	HPOS2:	DS	4	;HOME POSITION Y-AXIS
A35D	LSIMOD:	DS	1	;EXTENDED READ MODE
A35E	VTIME:	DS	2	;VELOCITY TIMER
A360	DSPOS1:	DS	4	;DISPLAY POSITION X-AXIS
A364	DSPOS2:	DS	4	;DISPLAY POSITION Y-AXIS
A368	DSVEL1:	DS	4	;DISPLAY VEL X-AXIS
A36C	DSVEL2:	DS	4	;DISPLAY VEL Y-AXIS
A370	FXRATE:	DS	4	;PARABOLIC FEEDRATE X-AXIS
A374	FYRATE:	DS	4	;PARABOLIC FEEDRATE Y-AXIS
A378	THETA:	DS	4	;CIRCLE ANGLE
A37C	DTHETA:	DS	6	;CIRCLE DESIRED ANGLE
A382	DDTHET:	DS	6	;CIRCLE DELTA ANGLE
A388	RADX:	DS	4	;X-AXIS RADIUS
A38C	RADY:	DS	4	;Y-AXIS RADIUS
A390	TCOUNT:	DS	2	
A392	ADIVP1:	DS	4	;AUTO DIVIDE POSITION X-AXIS
A396	ADIVP2:	DS	4	;AUTO DIVIDE POSITION Y-AXIS
A39A	ADCNT:	DS	2	;AUTO DIVIDE COUNT
A39C	LSTCTR:	DS	2	;EXTENDED READ COUNTER
A39E	LSTPTR:	DS	2	;EXTENDED READ POINTER
A3A0	EDPTR:	DS	2	;EDIT POINTER
A3A2	PDPTR:	DS	2	;PREVIOUS PROGRAM POINTER
A3A4	RAMEND:	DS	2	;END OF RAM ADDRESS
A3A6	DEPTH:	DS	1	;PLC DEPTH COUNTER
A3A7	FREE:	DS	6	;FREE SPACE
A3AD		DS	6	
A3B3	DTPTR:	DS	2	;PROGRAM POINTER
A3B5	STPTR:	DS	2	;STORAGE POINTER
A3B7	I60:	DS	1	;X-AXIS DEADBAND
A3B8	I61:	DS	1	;Y-AXIS DEADBAND
A3B9	IBITS4:	DS	1	
A3BA	I80:	DS	1	;PHASE LOCK GAIN
A3BB	I00:	DS	2	;FOLLOWING ERROR
A3BD	I01:	DS	2	;IN POSITION BAND
A3BF	I04:	DS	1	;HOME FEEDRATE
A3C0	I06:	DS	4	;SYSTEM FEEDRATE
A3C4	I08:	DS	2	;ACCEL TIME
A3C6	I10:	DS	2	;FOLLOWING ERROR WARNING
A3C8	I12:	DS	2	;MOVE DELAY TIME
A3CA	I17:	DS	4	;DISPLAY MODE

A3CE	I25:	DS	2	;X-AXIS HOME OFFSET
A3D0	I31:	DS	1	;X-AXIS JOG SPEED
A3D1	I34:	DS	2	;X-AXIS BACKLASH
A3D3	I35:	DS	4	;X-AXIS POSITIVE POSITION LIMIT
A3D7	I36:	DS	4	;X-AXIS NEGATIVE POSITION LIMIT
A3DB	I37:	DS	4	;X-AXIS POSITION RANGE
A3DF	I38:	DS	9	;X-AXIS POSITION FORMAT
A3E8	I51:	DS	1	;Y-AXIS JOG SPEED
A3E9	I54:	DS	2	;Y-AXIS BACKLASH
A3EB	I55:	DS	4	;Y-AXIS POSITIVE POSITION LIMIT
A3EF	I56:	DS	4	;Y-AXIS NEGATIVE POSITION LIMIT
A3F3	I57:	DS	4	;Y-AXIS POSITION RANGE
A3F7	I58:	DS	9	;Y-AXIS POSITION FORMAT
A400	I64:	DS	2	;X ENCODER/RESOLVER BASE ADDR.
A402	I65:	DS	2	;Y ENCODER/RESOLVER BASE ADDR.
A404	I66:	DS	1	;SPEED OF SECOND X RESOLVER
A405	I67:	DS	1	;SPEED OF SECOND Y RESOLVER
A406	I68:	DS	2	;SECOND X RESOLVER BASE ADDRESS
A408	I69:	DS	2	;SECOND Y RESOLVER BASE ADDRESS
A40A	I70:	DS	4	;X ENCODER/RESOLVER BIAS
A40E	I71:	DS	4	;Y ENCODER/RESOLVER BIAS
A412	I72:	DS	1	;X ENCODER/RESOLVER SIZE
A413	I73:	DS	1	;Y ENCODER/RESOLVER SIZE
A400	PSTART	EQU	0A400H	
A414	XSTART:	DS	1	;PROGRAM BUFFER START ADDRESS
C000	P8KEND	EQU	0C000H	
C000	EARTST	EQU	0C000H	
F800	P23END	EQU	0F800H	

Appendix E
SMCC E-Points and Test Points



E POINT	PHYSICAL LAYOUT	LOC.	DESCRIPTION	DEFAULT
E1	2 1 * *	D-8	JUMP Pin 1 to Pin 2 to increase gain of phase 1 output. No short is +/-5 volts out, short is +/- 10 volts out. See E9 also. Normally used at +/-10 volts out for single ended output and +/-5 volts out when output is used differentially.	Jumper installed +/-10 volts out
F2	1 2 * *	D-6	JUMP Pin 1 to Pin 2 for Sync/Out (use on A0 card only.) Note that if E2-1,2 is installed, E3-1,2 must be removed.	Jumper installed Card is set up as A0.
E3	1 2 * *	D-5	JUMP Pin 1 to Pin 2 for input sync (use on all cards other than A0). Also see E2.	No jumper (open)
E4	3 2 1 * * *	D-4	JUMP 1 to 2 to allow input bit from JEXP (J8 Pin 15) to be read by CPU. JUMP 2 to 3 to allow "servo 2" to go to JEXP (J8 Pin 15) to be used for R/D conversion.	No jumper (open)
E5		—	NOT USED	
E6	1 2 * *	D-4	JUMP Pin 1 to Pin 2. With jumper installed, EA/ (Pin 51) of CPU is grounded, which causes the CPU to use external memory. Otherwise, EA/ (Pin 51) is pulled up to +5 volts, CPU then uses internal memory only.	No jumper (open)
E7	1 2 * * * * 3 4	D-3	JUMP 1 to 2 to allow RXD/ as output J2 Pin a11 (J PAR-RS232) JUMP 3 to 4 to allow TXD/ as input J2 Pin a12 (J PAR-RS232) JUMP 1 to 3 to allow RXD/ as input J2 Pin a12 (J PAR-RS232) JUMP 2 to 4 to allow TXD/ as output J2 Pin a12 (J PAR-RS232)	1-2 jumper installed 3-4 jumper installed RXD/ is output TXD/ is input

E POINT	PHYSICAL LAYOUT	LOC.	DESCRIPTION	DEFAULT
E8	1 2 * * * * 3 4	D-3	JUMP 1 to 2 to allow RTS to be output, J2 pin a9 (J PAR-RS232) JUMP 3 to 4 to allow CTS to be input, J2 pin a10 (J PAR-RS232) JUMP 1 to 3 to allow RTS to be input, J2 PIN a9 (J PAR-RS232) JUMP 2 to 4 to allow CTS to be output, J2 PIN a10 (J PAR-RS232) Use E8 in conjunction with E7 for proper configuration.	1-2 jumper installed 3-4 jumper installed RTS is output CTS is input
E9	1 2 * *	C-7	JUMP Pin 1 to Pin 2 to increase gain of Phase 3 (See E1)	Jumper installed +/-10 volts out
E10		—	NOT USED	
E11	*1 *2	C-6	JUMP Pin 1 to Pin 2 to allow -14 VA to come from J11 (JPWR) (Ties Amplifier and SMCC power supply together. Defeats OPTO coupling Note that if E11 is changed, E15 and E24 must also be changed.	No jumper (open) No jumper
E12	*3 *2 *1	C-4	JUMP Pin 2 to Pin 3 to allow U17(Pin 26) SRAM and U17A (Pin 26) EAROM to input +5V. for 8KX8 parts. JUMP Pin 1 to Pin 2 to allow U17 (Pin 26) SRAM and U17A (Pin 26) EAROM to input A13 for 32KX8 parts.	1-2 jumper installed card is configured for 32KX8 parts.
E13	1 2 3 * * *	C-8	When short removed from 1 & 2 and placed on 2 and 3, provides X axis unipolar analog error output with direction bit on BSINX at J1C18. (See E18 for Y axis.)	1-2 jumper installed

Refer to Page E-1 for Location Map.

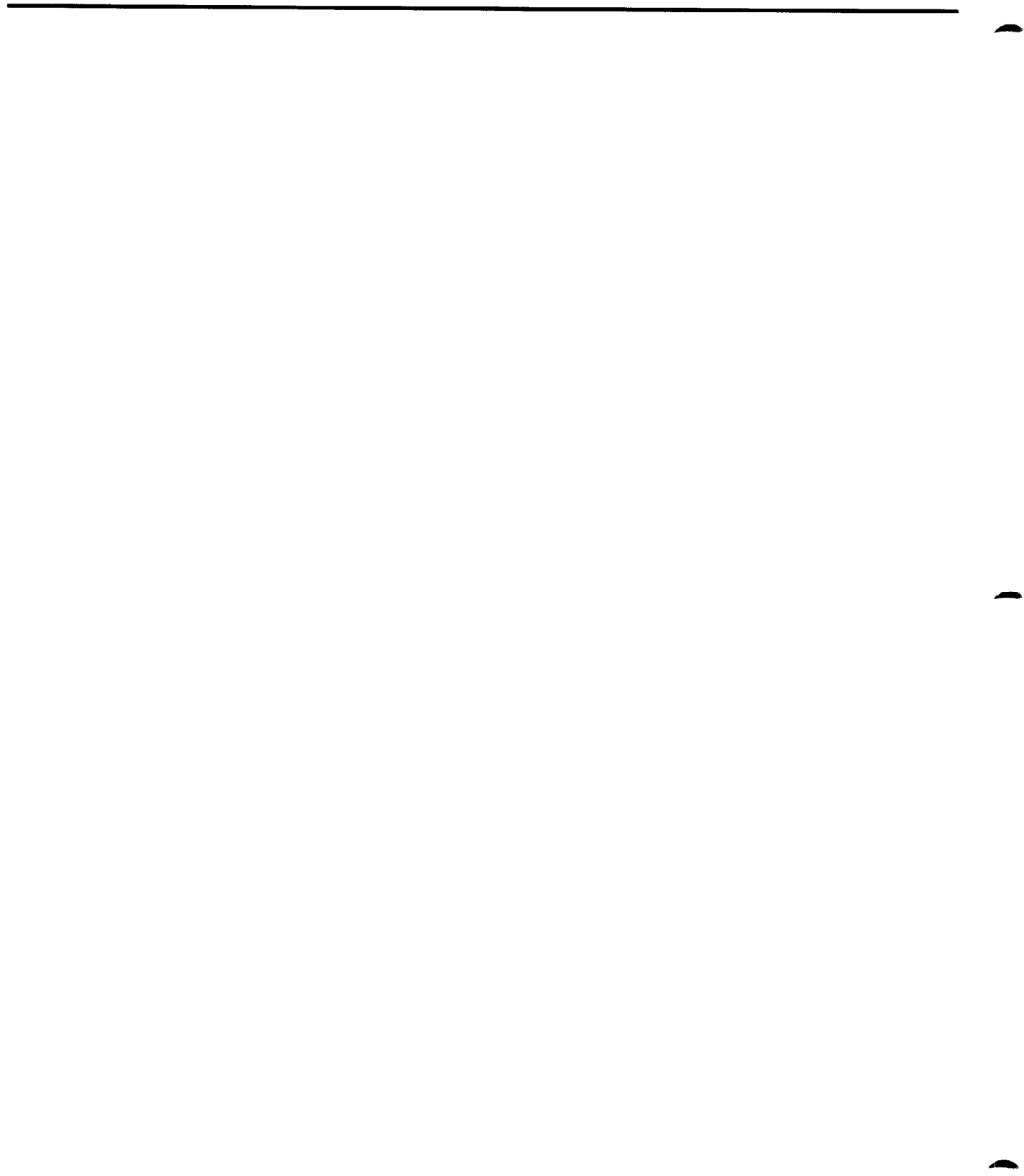
E POINT	PHYSICAL LAYOUT	LOC.	DESCRIPTION	DEFAULT
E14	3 2 1 * * *	C-7	JUMP Pin 1 to Pin 2 to allow AMPENA/signal (low true) JUMP Pin 2 to Pin 3 to allow AMPENA/signal (High true)	1-2 jumper installed
E15	*1 *2	C-7	JUMP Pin 1 to Pin 2 to allow analog GND to come from J PWR CONN J11 (Ties amplifier and SMCC GND together. Defeats OPTO coupling.) Note that if E15 is changed, E11 and E24 must also be changed.	No jumper (open)
E16		—	NOT USED	
E17		—	NOT USED	
E18	*1 *2 *3	B-8	When short removed from 1-2 and placed on 2-3, provides Y axis unipolar error output with direction bit on BSINY at J1A18. (See E13 for X axis)	1-2 installed
E19	2 1 * *	B-6	X axis selects the SIP Polarity 12T polarity selector open = positive input shorted = negative input	No jumper installed positive polarity
E20	2 1 * *	B-6	Y axis selects the SIP Polarity 12T polarity selector open = positive input shorted = negative input	No jumper installed positive polarity
E21	6* *1 5* *2 4* *3	B-5	JUMP Pin 1 to Pin 6 to send optically isolated BM01 to JMACH (J1) & JPWR (J11)	2-5 jumper installed IPOS/output
		B-4	JUMP Pin 2 to Pin 5 to send optically isolated BIPOS/ to go to JMACH (J1) & JPWR (J11)	
		B-4	JUMP Pin 3 to Pin 4 to send optically isolated BIPOS to go to JMACH (J1) & JPWR (J11)	

E POINT	PHYSICAL LAYOUT	LOC.	DESCRIPTION	DEFAULT
E22	1 2 * *	B-5	When installed, allows soft following error (i10) to also drive the Following error output, FEOP/ (J1-3)	No jumper installed. Only i00 drives the FEOP/output.
E23	1 2 3 * * *	B-4	JUMP Pin 2 to Pin 3 to allow phasing without hall effects. JUMP Pin 1 to 2 to phase with hall effects.	2-3 jumper installed
E24	*2 *1	A-5	JUMP Pin 1 to Pin 2 to allow +14 VA to come from JPWR CONN. J11 (Ties Amplifier and SMCC power supply together. Defeats OPTO-coupling). Note that if E24 is changed, E11 and E15 must also be changed.	No jumper (open)
E25	3 2 1 * * *	A-3	JUMP Pin 1 to Pin 2 to obtain +5V to top of P01, R17 (5K) for VEE of LCD display's positive bias. JUMP Pin 3 to Pin 2 to obtain -5V to top of P01, R17 (5K) for VEE of LCD display's negative bias.	1-2 jumper installed (positive bias)
E26	*1 *2	A-3	JUMP Pin 1 to Pin 2 to allow +5V to go to J2 Pin a20 & c13 (JPWR-RS232) CONN.	No jumper installed
E27	*3 *2 *1	A-1	JUMP Pin 1 to Pin 2 to connect A5 to R/W displays & JPWR/RS232.	2-3 jumper installed
E28 - E39		—	NOT USED	
E40	2 1 * *	C-6	JUMP Pin 1 to Pin 2 to obtain normal X encoder operation. Remove Jumper for digital tach (1/T). X encoder operation.	1-2 jumper installed (X encoder not 1/T)
E41	1 2 * *	C-6	Same as E40, but for Y axis.	1-2 jumper installed (Y encoder not 1/T)

E POINT	PHYSICAL LAYOUT	LOC.	DESCRIPTION	DEFAULT
E42	*1 *2 *3	A-6	JUMP Pin 1 to Pin 2 to obtain differential X position encoder input mode. JUMP Pin 2 to Pin 3 to obtain non-differential X encoder input mode, this will bias X encoder negative inputs to 1/2 VCC = 2.5 V. (See E50)	2-3 jumper installed (X encoder non-diff.)
E43		—	NOT USED	
E44		—	NOT USED	
E45	*1 *2 *3	B-3	JUMP Pin 1 to Pin 2 to obtain differential X handwheel input mode. JUMP Pin 2 to Pin 3 to obtain non-differential X handwheel input mode, this will bias X-HW negative inputs to 1/2 VCC = 2.5V	2-3 jumper installed (X - HW enc non-diff.)
E46	*1 *2 *3	C-3	Same as E45, but for Y handwheel.	2-3 jumper installed (Y - HW enc non-diff.)
E47	2 1 * *	B-3	Jumper Pin 1 to Pin 2 to tie X-handwheel and Y-handwheel together for input from a front panel at J7-16 for Chan.A.(Same as E48 for Chan B)	No jumper
E48	*1 *2	B-1	Jumper Pin 1 to Pin 2 to tie X-handwheel and Y-handwheel together for input from a front panel at J7-22 for Chan.B. (Same as E47 for Chan A) Note: Both E47 and E48 must be changed for proper operation.	No jumper
E49		—	NOT USED	
E50	*3 *2 *1	A-7	JUMP Pin 1 to Pin 2 to obtain differential Y Position encoder input mode. (See E42)	2-3 jumper installed

TEST POINTS		
TEST POINT	LOCATION	DESCRIPTION
TP1	D-7	Watchdog timer test point, Low = O.K. High = watchdog Timer time out, SMCC not functioning.
TP2	D-6	12 MHZ Clock
TP3	B-7	YENCA TP (Y ENCODER CHANNEL A TP)
TP4	B-6	XENCB TP (X ENCODER CHANNEL B TP)
TP5	B-6	XENCA TP (X ENCODER CHANNEL A TP)
TP6	B-7	YENCB TP (Y ENCODER CHANNEL B TP)
TP7	A-6	YENCC TP (Y ENCODER CHANNEL C TP)
TP8	A-8	PHASE 1 TP = X AXIS COMMAND SIGNAL or X AXIS PHASE 1 SIGNAL
TP9	A-8	PHASE 3 TP = Y AXIS COMMAND SIGNAL or X AXIS PHASE 3 SIGNAL
TP10	A-6	XENCC 1P (X ENCODER CHANNEL C TP)
TP11	A-5	INIT/ RESET INPUT TP, LOW = RESET CONDITION HIGH= NON-RESET CONDITION

Refer to Page E-1 for Location Map.



Appendix F

SMCC Connectors and Pin Assignments

SMCC CONNECTORS	LOC.	LIST OF MATING CONNECTIONS
J1 (JMACH)	A-8	AMLAN P/N C96F3LA+B+C
J2 (JPAN-RS232)	A-1	AMLAN P/N C64F3SCHM.Z.A+C 3M P/N 6964-0000-EF OR equivalent
J3	—	NOT USED
J4	—	NOT USED
J5 (JDISPX)	A-2	H and T P/N 4003-14-00-P5 3M P/N 3385-6600 or equivalent
J6 (JDISPY)	A-2	H and T P/N 4003-14-00-P5 3M P/N 3385-6600 or equivalent
J7 (JPAN)	A-7	H and T P/N 4003-26-00-P5 3M P/N 3399-6600 or equivalent
J8 (JEXP)	D-4	H and T P/N 4003-20-00-P5 3M P/N 3421-6600 or equivalent
J9 (JXHW)	A-1	H and T P/N 4003-10-00 P5 3M P/N 3473-6600 or equivalent
J10 (POWER)	D-1	NONE
J11 (JYHW)	A-1	H and T P/N 4003-10-00-P5 3M P/N 3473-6600 or equivalent
J12 (JOPT)	A 3	H and T P/N 4003-26-00-P5 3M P/N 3399-6600 or equivalent

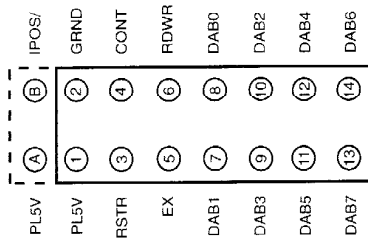
Refer to Page E-1 for Location Map.

A	B	C
32 FALT/	32 XJMI/	32 FALT/
31 AENA	31 YJMI/	31 AENA
30 AGND	30 XJPL/	30 AGND
29 P15A	29 YJPL/	29 P15A
28 MAI5/	28 STRT/	28 MAI5/
27 MAI6/	27 PREJ/	27 MAI6/
26 MAI7/	26 STOP/	26 MAI7/
25 M15A	25 STEP/	25 M15A
24 PHA3	24 HOLD/	24 PHA1
23 PHA3/	23 HOME/	23 PHA1/
22 SPAR	22 HWXD/	22 SPAR
21 BPO5/	21 HWYD/	21 BPO1
20 BPO6/	20 SPAR	20 BPO2/
19 BPO4/	19 INIT/	19 BPO3/
18 BSNY	18 BFLD/	18 BSNX
17 CHAY	17 IPLD/	17 CHAX
16 CHAY/	16 EPLD/	16 CHAX/
15 CHBY	15 ERLD/	15 CHBX
14 CHBY/	14 SPAR	14 CHBX/
13 CHCY	13 ITLD/	13 CHCX
12 CHCY/	12 F2LD/	12 CHCX/
11 YHFL/	11 F1LD/	11 XHFL/
10 YUFL/	10 P15V	10 XUFL/
9 SPAR	9 M15V	9 SPAR
8 GRND	8 GRND	8 GRND
7 GRND	7 GRND	7 GRND
6 PL5V	6 PL5V	6 PL5V
5 YMIL	5 SPAR	5 XMIL
4 YPLL	4 IPRT	4 XPLL
3 CUR1	3 FERT	3 CURO
2 AVSS	2 FEOP/	2 AVSS
1 ANA3	1 IPOP/	1 ANA2

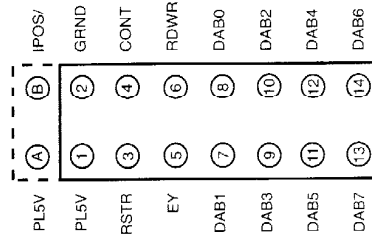
J1

A	B	C			
32	STRB/	32	MAI7/	32	GRND
31	DATO	31	MAI6/	31	SELO
30	DAT1	30	MAI5/	30	SEL1
29	DAT2	29	MAI4/	29	SEL2
28	DAT3	28	MAI3/	28	SEL3
27	DAT4	27	MAI2/	27	SEL4
26	DAT5	26	MAI1/	26	SEL5
25	DAT6	25	MAO5/	25	SEL6
24	DAT7	24	MAO4/	24	SEL7
23	ACKN/	23	MAO3/	23	GRND
22	BUFU	22	MAO2/	22	GRND
21	IPOS	21	MAO1/	21	GRND
20	SP5V	20	PL5V	20	INIT/
19	GRND	19	EY	19	EROF/
18	IPOS	18	CONT	18	GRND
17	GRND	17	RSTR	17	ORDY/
16	CHASS	16	RDWR	16	HSB/
15	BUFU/	15	EX	15	SYNC/
14	SPAR	14	DAB0	14	SPAR
13	CHAS	13	DAB1	13	SP5V
12	TXD/	12	DAB2	12	ENBY/
11	RXD/	11	DAB3	11	ENAY/
10	RETS	10	DAB4	10	ENBX
9	CLTS	9	DAB5	9	ENAX
8	DASR	8	DAB6	8	SPAR
7	GRND	7	DAB7	7	DATR
6	BUFU/	6	HWYD/	6	SPAR
5	EROF/	5	HWXD/	5	IPOS
4	INIT/	4	ENBY	4	BTXD
3	GRND	3	ENAY	3	BRTS
2	GRND	2	ENBX	2	SYNC/
1	GRND	1	ENAX	1	PL5V

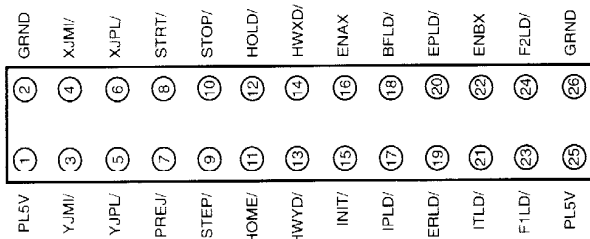
J2



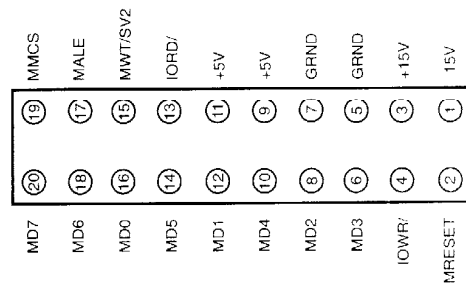
J5



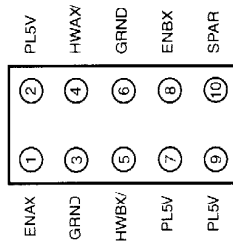
J6



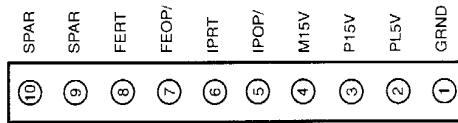
J7



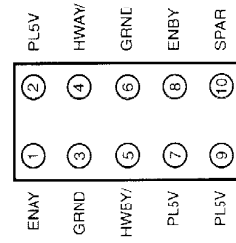
J8



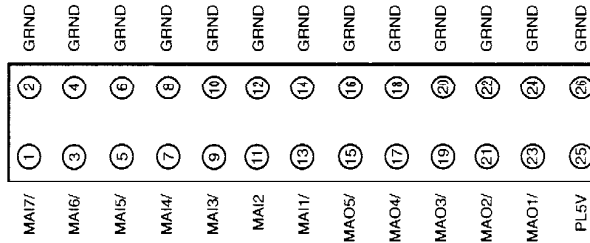
J9



J10



J11



J12

SMCC CONNECTORS LISTING

J1 (JMACH) Connector (96 Pin DIN Connector)				
PIN #	SYMBOL	NOTES	FUNCTION	DESCRIPTION
Row "A"				
A1	ANA3	INPUT	ANALOG INPUT 3	Could be used as analog speed command. Use 0 to +5V for 0 to full scale.
A2	AVSS	INPUT	ANALOG RETURN	
A3	CUR1	INPUT	CURRENT 1 FEEDBACK	
A4	YPLL	INPUT	Y AXIS + LIMIT SW.	Current limiting
A5	YMIL	INPUT	Y AXIS - LIMIT SW	Low enables move
A6	PL5V	I or 0	+5VDC POWER SUPPLY	Low enables move
A7	GRND	I or 0	+5 VDC RETURN	Could be used as input to card or output to encoders.
A8	GRND	I or 0	+5 VDC RETURN	Logic GND
A9	SPAR	SPARE		Logic GND
A10	YUFL/	INPUT	Y AXIS USER FLAG	Programmable polarity
A11	YHFL/	INPUT	Y AXIS HOME FLAG	Programmable polarity
A12	CHCY/	INPUT	ENCODER C CH. NEG	Y axis (Do not GND if not used)
A13	CHCY	INPUT	ENCODER C CH. POS	Y axis
A14	CHBY/	INPUT	ENCODER B CH. NEG	Y axis (Do not GND if not used)
A15	CHBY	INPUT	ENCODER B CH. POS	Y axis
A16	CHAY/	INPUT	ENCODER A CH. NEG	Y axis (Do not GND if not used)
A17	CHAY	INPUT	ENCODER A CH. POS	Y axis
A18	BSNY	OUTPUT	Y PWM SIGN	Programmable level
A19	BPO4/	OUTPUT	PWM OUT	Nonbuffered pulse
A20	BPO6/	OUTPUT	PWM OUT	PHASE 1, 2, 3, or 4
A21	BPO5/	OUTPUT	PWM OUT	
A22	SPAR	SPARE		
A23*	PHA3/	OUTPUT	PHASE 3 RETURN	Analogue differential
A24	PHA3	OUTPUT	PHASE 3 COMMAND	Signal Out Y axis
A25	M15A	INPUT	-15V SUPPLY	From power amp
A26	MA17/	INPUT	MACHINE INPUT 7	Low is a True
A27	MA16/	INPUT	MACHINE INPUT 6	Low is a True
A28	MA15/	INPUT	MACHINE INPUT 5	Low is a True
A29	P15A	INPUT	+15V SUPPLY	From power amp
A30	AGND	INPUT	+/- 15V COMMON	From power amp
A31	AENA	OUTPUT	AMPLIFIER ENABLE	Programmable polarity
A32	FALT/	INPUT	AMPLIFIER FAULT	Low to A30 is Fault

***NOTE:** If PHA3/ is not needed for the control, do not tie to AMP-common(AGND).

J1 (JMACH) Connector (96 Pin DIN Connector)				
PIN #	SYMBOL NOTES		FUNCTION	DESCRIPTION
Row "B"				
B1	IPOP/	OUTPUT	(see NOTE 1)	
B2	FEOP/	OUTPUT	FOL.ERR OPTO.ISOLATED	
B3	FERT	OUTPUT	ERROR RETURN	Transistor Emitter
B4	IPRT	OUTPUT	IN POSITION RETURN	Transistor Emitter
B5	SPAR	SPARE		
B6	PL5V	INPUT	+5 VDC POWER SUPPLY	
B7	GRND	INPUT	5 VDC COMMON	
B8	GRND	INPUT	5 VDC COMMON	
B9	M15V	INPUT	-15 V POWER SUPPLY	
B10	P15V	INPUT	+15 V POWER SUPPLY	
B11	F1LD/	OUTPUT	FOLLOWING ERROR 1	Low Lights LED
B12	F2LD/	OUTPUT	FOLLOWING ERROR 2	Low Lights LED
B13	ITLD/	OUTPUT	CURRENT LIMIT IND.	Low Lights LED
B14	SPAR	SPARE		
B15	ERLD/	OUTPUT	ERROR INDICATOR	Low Lights LED
B16	EPLD/	OUTPUT	END OF PROGRAM IND.	Low Lights LED
B17	IPLD/	OUTPUT	IN POSITION IND.	Low Lights LED
B18	BFLD/	OUTPUT	BUFFER FULL IND.	Low Lights LED
R19	INIT/	INPUT	RESETS SMCC	Low is a Reset
B20	SPAR	SPARE		
B21	HWYD/	INPUT	HANDWH. Y DISABLE	Low disables
B22	HWXD/	INPUT	HANDWH. X DISABLE	Low disables
B23	HOME/	NPUT	GO HOME COMMAND	Low is GO HOME
B24	HOLD/	INPUT	HOLD MOTION	Low is Hold
B25	STEP/	INPUT	STEP THROUGH PROGRAM	Low is Step
R26	STOP/	INPUT	STOP PROGRAM RUN	Low is Stop
B27	PREJ/	INPUT	RET. TO PREJOG POS.	Low is Return
B28	STRT/	INPUT	START PROGRAM RUN	Low is Start
B29	YJPL/	INPUT	JOG Y AXIS IN +DIR.	Low is Jog Y+
B30	XJPL/	INPUT	JOG X AXIS IN +DIR.	Low is Jog X+
B31	YJMI/	INPUT	JOG Y AXIS IN -DIR.	Low is Jog Y-
B32	XJMI/	INPUT	JOG X AXIS IN -DIR.	Low is Jog X-

NOTE 1: Optically isolated IPOP/ function could be changed by E21 to the following:

- In position for this card "Low is true".
- In position for this card "High is true".
- MO1/output "Low is true".

J1 (JMACH) Connector (96 Pin DIN Connector)

PIN #	SYMBOL NOTES		FUNCTION	DESCRIPTION
Row "C"				
C1	ANA2	INPUT	ANALOG INPUT 2	Could be used as an analog speed command Use 0 to +5V for 0 to full scale.
C2	AVSS	INPUT	ANALOG RETURN	Current Limiting.
C3	CUR0	INPUT	CURRENT 0 FEEDBACK	Low enables move
C4	XPLL	INPUT	X AXIS + LIMIT SW	Low enables move
C5	XMIL	INPUT	X AXIS - LIMIT SW	Low enables move
C6	PL5V	I or 0	+5 VDC POWER SUPPLY	Could be used as an input to card or output to encoders.
C7	GRND	I or 0	+5 VDC RETURN	Logic GND
C8	GRND	I or 0	+5 VDC RETURN	Logic GND
C9	SPAR	SPARE		
C10	XUFL/	INPUT	X AXIS USER FLAG	Programmable polarity
C11	XHFL/	INPUT	X AXIS HOME FLAG	Programmable polarity
C12	CHCX/	INPUT	ENCODER C CH. NEG	X Axis (Do not GND if not used)
C13	CHCX	INPUT	ENCODER C CH. POS	X Axis
C14	CHBX/	INPUT	ENCODER B CH. NEG	X Axis (Do not GND if not used)
C15	CHBX	INPUT	ENCODER B CH. POS	X Axis
C16	CHAX/	INPUT	ENCODER A CH. NEG	X Axis (Do not GND if not used)
C17	CHAX	INPUT	ENCODER A CH. POS	X Axis
C18	BSNX	OUTPUT	X PWM SIGN	Programmable level
C19	BPO3/	OUTPUT	PWM OUT	Nonbuffered pulse PHASE 1, 2, 3, or 4
C20	BPO2/	OUTPUT	PWM OUT	
C21	BPO1/	OUTPUT	PWM OUT	
C22	SPAR	SPARE		
C23*	PHA1/	OUTPUT	PHASE 1 RETURN	Analogue differential
C24	PHA1	OUTPUT	PHASE 1 RETURN	signal out X axis
C25	M15A	INPUT	-15 V SUPPLY	From power amp
C26	MAI7/	INPUT	MACHINE INPUT 7	Low is a True
C27	MAI6/	INPUT	MACHINE INPUT 6	Low is a True
C28	MAI5/	INPUT	MACHINE INPUT 5	Low is a True
C29	P15A	INPUT	+15 V SUPPLY	From power amp
C30	AGND	INPUT	+/- 15 V COMMON	From power amp
C31	AENA	OUTPUT	AMPLIFIER ENABLE	Programmable polarity
C32	FALT/	INPUT	AMPLIFIER FAULT	Low to C30 is Fault

***NOTE:** If PHA1/ is not needed for Servo Amplifier, do not tie to Amp-Common (AGND)

J2 JPAR-RS232 Connector (96 Pin DIN Connector)				
PIN #	SYMBOL NOTES		FUNCTION	DESCRIPTION
Row "A"				
A1	GRND	COMMON	SMCC COMMON	
A2	GRND	COMMON	SMCC COMMON	
A3	GRND	COMMON	SMCC COMMON	
A4	INIT/	INPUT	RESETS SMCC	Low is a "Reset"
A5	EROR/	OUTPUT	SIGNALS ERROR	Low is an "Error"
A6	BUFU/	OUTPUT	BUFFER FULL	Low is "Buffer full"
A7	GRND	COMMON	SMCC COMMON	
A8	DASR	RS 232	DATA SET READY	
A9	CLTS	RS 232	CLEAR TO SEND	9A & 10A functions
A10	RETS	RS 232	REQUEST TO SEND	E8 could exchange
A11	RXD/	RS 232	RECEIVE DATA	E7 could exchange
A12	TXD/	RS 232	TRANSMIT DATA	11A&12A functions
A13	CHAS		CHASSIS	Not tied to common
A14	SPAR	SPARE		
A15	BUFU/	OUTPUT	BUFFER FULL	Low is "Buffer full"
A16	CHAS		CHASSIS	Not tied to common
A17	GRND	COMMON	SMCC COMMON	
A18	IPOS	OUTPUT	IN POSITION	High is "In position"
A19	GRND	COMMON	SMCC COMMON	
A20	SP5V	OUTPUT	+5 VDC SUPPLY	Deactivated by E26
A21	IPOS/	OUTPUT	IN POSITION	Low is "In position"
A22	BUFU	OUTPUT	BUFFER FULL	High is buffer full
A23	ACKN/	OUTPUT	HANDSHAKING	Low is handshake
A24	DAT7	BIDIRECT	DATA 7	Parallel Interface
A25	DAT6	BIDIRECT	DATA 6	Parallel Interface
A26	DAT5	BIDIRECT	DATA 5	Parallel Interface
A27	DAT4	BIDIRECT	DATA 4	Parallel Interface
A28	DAT3	BIDIRECT	DATA 3	Parallel Interface
A29	DAT2	BIDIRECT	DATA 2	Parallel Interface
A30	DAT1	BIDIRECT	DATA 1	Parallel Interface
A31	DAT0	BIDIRECT	DATA 0	Parallel Interface
A32	STRB/	INPUT	STROBE	Low is a "Strobe"

J2 JPAR-RS232 Connector (96 Pin DIN Connector)

PIN #	SYMBOL NOTES		FUNCTION	DESCRIPTION
Row "B"				
B1	ENAX	INPUT	X HANDWH ENC.A CH.	5V TTL sqr pulse
B2	ENBX	INPUT	X HANDWH ENC.B CH.	5V TTL sqr pulse
B3	ENAY	INPUT	Y HANDWH ENC.A CH.	5V TTL sqr pulse
B4	ENBY	INPUT	Y HANDWH ENC.B CH.	5V TTL sqr pulse
B5	HWXD/	INPUT	X HANDWH DISABLE	Low is "Disable"
B6	HWYD/	INPUT	Y HANDWH DISABLE	Low is "Disable"
B7	DAB7	BIDIRECT	DISPLAY DATA 7	5V TTL
B8	DAB6	BIDIRECT	DISPLAY DATA 6	5V TTL
B9	DAB5	BIDIRECT	DISPLAY DATA 5	5V TTL
B10	DAB4	BIDIRECT	DISPLAY DATA 4	5V TTL
B11	DAB3	BIDIRECT	DISPLAY DATA 3	5V TTL
B12	DAB2	BIDIRECT	DISPLAY DATA 2	5V TTL
B13	DAB1	BIDIRECT	DISPLAY DATA 1	5V TTL
B14	DAB0	BIDIRECT	DISPLAY DATA 0	5V TTL
B15	EX	OUTPUT	X DISPLAY ENABLE	High is "Enable"
B16	RDWR	BIDIRECT	READ OR WRITE	E27 selects function
B17	RSTR	OUTPUT	READ STROBE	TTL
B18	CONT	OUTPUT	CONTRAST ADJUST	0 to 5 VDC
B19	EY	OUTPUT	Y DISPLAY ENABLE	High is "Enable"
B20	PL5V	OUTPUT	+5V	Power supply out
B21	MAO1/	OUTPUT	MACHINE OUTPUT 1	Open collector TTL
B22	MAO2/	OUTPUT	MACHINE OUTPUT 2	Low is a true
B23	MAO3/	OUTPUT	MACHINE OUTPUT 3	Low is a true
B24	MAO4/	OUTPUT	MACHINE OUTPUT 4	Low is a true
B25	MAO5/	OUTPUT	MACHINE OUTPUT 5	Low is a true
B26	MAI1/	INPUT	MACHINE INPUT 1	5V TTL, High is true
B27	MAI2/	INPUT	MACHINE INPUT 2	5V TTL, High is true
B28	MAI3/	INPUT	MACHINE INPUT 3	5V TTL, High is true
B29	MAI4/	INPUT	MACHINE INPUT 4	5V TTL, High is true
B30	MAI5/	INPUT	MACHINE INPUT 5	5V TTL, Low is true
B31	MAI6/	INPUT	MACHINE INPUT 6	5V TTL, Low is true
B32	MAI7/	INPUT	MACHINE INPUT 7	5V TTL, Low is true

J2 JPAR-RS232 Connector (96 Pin DIN Connector)				
PIN #	SYMBOL NOTES		FUNCTION	DESCRIPTION
Row "C"				
C1	PL5V	OUTPUT	+5V DC	Power supply out
C2	SYNC/	OUTPUT	SYNCHRONIZER	Low True from A0
C3	BRTS	OUTPUT	BUFFERED RTS	RS232 daisy-chain
C4	BTXD	OUTPUT	BUFFERED TXD	RS232 daisy-chain
C5	IPOS	OUTPUT	IN POSITION	High "In Position"
C6	SPAR	SPARE		
C7	DATR	RS232	DATA TERMINAL READY	
C8	SPAR	SPARE		
C9	ENAX/	INPUT	X HWL ENC. A CH NEG	5V TTL
C10	ENBX/	INPUT	X HWL ENC. B CH NEG	5V TTL
C11	ENAY/	INPUT	Y HWL ENC. A CH NEG	5V TTL
C12	ENBY/	INPUT	Y I IWL ENC. B CH NEG	5V TTL
C13	SP5V	OUTPUT	+5 VDC SUPPLY	Deactivated by E26
C14	SPAR	SPARE		
C15	SYNC/	OUTPUT	SYNCHRONIZER	Low true from A0
C16	HSB/	INPUT	READ STROBE	Low "Read Strobe"
C17	ORDY/	OUTPUT	OUTPUT READY	Low "Read Strobe"
C18	GRND	COMMON	SMCC COMMON	Low "Read Strobe:
C19	EROR/	OUTPUT	SIGNALS ERROR	Low "Error"
C20	INIT/	INPUT	RESETS SMCC	Low "Reset"
C21	GRND	COMMON	SMCC COMMON	
C22	GRND	COMMON	SMCC COMMON	
C23	GRND	COMMON	SMCC COMMON	
C24	SEL7	OUTPUT	DATA7	Scanner output for reading thumbwheel switches
C25	SEL6	OUTPUT	DATA6 OUTPUT	" " "
C26	SEL5	OUTPUT	DATA5 OUTPUT	" " "
C27	SEL4	OUTPUT	DATA4 OUTPUT	" " "
C28	SEL3	OUTPUT	DATA3 OUTPUT	" " "
C29	SEL2	OUTPUT	DATA2 OUTPUT	" " "
C30	SEL1	OUTPUT	DATA1 OUTPUT	" " "
C31	SEL0	OUTPUT	DATA0 OUTPUT	" " "
C32	GRND	COMMON	SMCC COMMON	

J5 JDISPX Connector (14 Pin Header)				
PIN #	SYMBOL NOTES		FUNCTION	DESCRIPTION
1	PL5V	OUTPUT	+5V POWER	TTL 0 to 10 VDC High is "Enable" E27 selects func. 5V TTL 5V TTL 5V TTL 5V TTL 5V TTL 5V TTL 5V TTL 5V TTL 5V TTL
2	GRND	COMMON	SMCC COMMON	
3	RSTR	OUTPUT	READ STROBE	
4	CONT	OUTPUT	CONTRAST ADJUST.VEE	
5	EX	OUTPUT	X DISPLAY ENABLE	
6	RDWR	BIDIRECT	READ OR WRITE	
7	DAB1	BIDIRECT	DISPLAY DATA1	
8	DAB0	BIDIRECT	DISPLAY DATA0	
9	DAB3	BIDIRECT	DISPLAY DATA3	
10	DAB2	BIDIRECT	DISPLAY DATA2	
11	DAB5	BIDIRECT	DISPLAY DATA5	
12	DAB4	BIDIRECT	DISPLAY DATA4	
13	DAB7	BIDIRECT	DISPLAY DATA7	
14	DAB6	BIDIRECT	DISPLAY DATA6	

J6 JDISPY Connector (14 Pin Header)				
PIN #	SYMBOL NOTES		FUNCTION	DESCRIPTION
1	PL5V	OUTPUT	+5V POWER	TTL 0 to 10 VDC High is "Enable" E27 selects func. 5V TTL 5V TTL 5V TTL 5V TTL 5V TTL 5V TTL 5V TTL 5V TTL 5V TTL
2	GRND	COMMON	SMCC COMMON	
3	RSTR	OUTPUT	READ STROBE	
4	CONT	OUTPUT	CONTRAST ADJUST	
5	EY	OUTPUT	Y DISPLAY ENABLE	
6	RDWR	BIDIRECT	READ OR WRITE	
7	DAB1	BIDIRECT	DISPLAY DATA1	
8	DAB0	BIDIRECT	DISPLAY DATA0	
9	DAB3	BIDIRECT	DISPLAY DATA3	
10	DAB2	BIDIRECT	DISPLAY DATA2	
11	DAB5	BIDIRECT	DISPLAY DATA5	
12	DAB4	BIDIRECT	DISPLAY DATA4	
13	DAB7	BIDIRECT	DISPLAY DATA7	
14	DAB6	BIDIRECT	DISPLAY DATA6	

J7 JPAN Connector (26 Pin Header)				
PIN #	SYMBOL NOTES		FUNCTION	DESCRIPTION
1	PL5V	OUTPUT	+5V POWER	For remote panel
2	GRND	COMMON	SMCC COMMON	
3	YJMI/	INPUT	JOG Y AXIS IN -DIR	Low is Jog Y-
4	XJMI/	INPUT	JOG X AXIS IN -DIR	Low is Jog X-
5	YJPL/	INPUT	JOG Y AXIS IN +DIR	Low is Jog Y+
6	XJPL/	INPUT	JOG X AXIS IN +DIR	Low is Jog X+
7	PREJ/	INPUT	RET. TO PREJOG POS.	Low is Return
8	STRT/	INPUT	START PROGRAM RUN	Low is Start
9	STEP/	INPUT	STEP THRU PROGRAM	Low is Step
10	STOP/	INPUT	STOP PROGRAM RUN	Low is Stop
11	HOME/	INPUT	GO HOME COMMAND	Low is Go Home
12	HOLD/	INPUT	HOLD MOTION	Low is Hold
13	HWYD/	INPUT	HANDWH. Y DISABLE	Low disables
14	HWXD/	INPUT	HANDWH. X DISABLE	Low disables
15	INIT/	INPUT	RESETS SMCC	Low is a Reset
16	ENAX	INPUT	X HANDWH ENC. A CH.	5V TTL sq.pulse
17	IPLD/	OUTPUT	IN POSITION IND.	Low lights LED
18	BFLD/	OUTPUT	BUFFER FULL IND.	Low lights LED
19	ERLD/	OUTPUT	ERROR INDICATOR	Low lights LED
20	EPLD/	OUTPUT	END OF PROGRAM IND.	Low lights LED
21	ITLD/	OUTPUT	CURRENT LIMIT IND.	Low lights LED
22	ENBX/	OUTPUT	Y HANDWH ENC. B CH.	5V TTL sq. pulse
23	FILD/	OUTPUT	FOLLOWING ERROR 1	5V TTL sq. pulse
24	F2LD/	OUTPUT	FOLLOWING ERROR 2	5V TTL sq. pulse
25	PL5V	OUTPUT	+5V POWER	For remote panel
26	GRND	COMMON	SMCC COMMON	

J8 JEXP Connector (20 Pin Header)				
PIN #	SYMBOL NOTES		FUNCTION	DESCRIPTION
1	-15V	OUTPUT	-15V POWER	For extension card For extension card For extension card Low is Write
2	M RESET	OUTPUT	RESET	
3	+15V	OUTPUT	+15V POWER	
4	IOWR/	OUTPUT	WRITE DATA	
5	GRND	OUTPUT	SMCC COMMON	
6	MD3	BIDIRECT	ADDRESS DATA3	
7	GRND	COMMON	SMCC COMMON	
8	MD2	BIDIRECT	ADDRESS DATA2	
9	+5V	OUTPUT	+5V POWER	
10	MD4	BIDIRECT	ADDRESS DATA4	
11	+5V	OUTPUT	+5V POWER	Low is Read (Not used in SMCC) (Controlled by E4)
12	MD1	BIDIRECT	ADDRESS DATA1	
13	IORD/	OUTPUT	READ DATA	
14	MD5	BIDIRECT	ADDRESS DATA5	
15	MWT/SV2	INPUT/ OUTPUT	GENERAL INPUT/ SERVO OUTPUT 2	
16	MD0	BIDIRECT	ADDRESS DATA0	
17	MALE	OUTPUT	ADDRESS LATCH ENABLE	
18	MD6	BIDIRECT	ADDRESS DATA6	
19	MMCS	OUTPUT	BOARD SELECT	
20	MD7	BIDIRECT	ADDRESS DATA7	

J9 JXHW Connector (10 Pin Header)				
PIN #	SYMBOL NOTES		FUNCTION	DESCRIPTION
1	ENAX	INPUT	X HANDWHEEL A CH.	5V TTL sq. pulse
2	PL5V	OUTPUT	+5V POWER	
3	GRND	COMMON	SMCC COMMON	
4	HWAX/	INPUT	X HNDWH. A CH. NEG.	5V TTL sq. pulse
5	HWBX/	INPUT	X HNDWH. B CH. NEG.	
6	GRND	COMMON	SMCC COMMON	5V TTL sq. pulse
7	PL5V	OUTPUT	+5V POWER	
8	ENBX	INPUT	X HANDWHEEL B CH.	
9	PL5V	OUTPUT	+5V POWER	
10	SPAR	SPARE		

J10 Power Connector (10 Pin Terminal Strip)				
PIN #	SYMBOL NOTES		FUNCTION	DESCRIPTION
1	GRND	INPUT	5V DC COMMON	Power input
2	PL5V	INPUT	+5V DC SUPPLY	Power input
3	P15V	INPUT	+15V DC SUPPLY	Power input
4	M15V	INPUT	-15V DC SUPPLY	Power input
5	IPOP/	OUTPUT	MULTIPLE FUNCTION	Opto-isolated
6	IPRT	OUTPUT		Opto-isolated
7	FEOP	OUTPUT	FOLLOWING ERROR FOLL. ERROR RETURN	Opto-isolated
8	FERT	OUTPUT		Opto-isolated
9	SPAR	SPARE		
10	SPAR	SPARE		

J11 JYHW Connector (10 Pin Header)				
PIN #	SYMBOL NOTES		FUNCTION	DESCRIPTION
1	ENAY	INPUT	Y HANDWHEEL A CH. I.	5V TTL sq. pulse
2	PL5V	OUTPUT	+5V POWER	
3	GRND	COMMON	SMCC COMMON	5V TTL sq. pulse
4	HWAY/	INPUT	Y HANDWHEEL A CH.	
5	IWBV/	INPUT	Y HANDWHEEL B CH.	5V TTL sq. pulse
6	GRND	COMMON	SMCC COMMON	5V TTL sq. pulse
7	PL5V	OUTPUT	+5V POWER	
8	ENBY	INPUT	Y HANDWHEEL B CH.	
9	PL5V	OUTPUT	+5V POWER	
10	SPAR	SPARE		

J12 JOPT Connector (26 Pin Header)				
PIN #	SYMBOL NOTES		FUNCTION	DESCRIPTION
1	MAI7/	INPUT	MACHINE INPUT 7	5V TTL, High is False
2	GRND	COMMON	SMCC COMMON	
3	MAI6/	INPUT	MACHINE INPUT 6	5V TTL, High is False
4	GRND	COMMON	SMCC COMMON	
5	MAI5/	INPUT	MACHINE INPUT 5	5V TTL, High is False
6	GRND	COMMON	SMCC COMMON	
7	MAI4/	INPUT	MACHINE INPUT 4	5V TTL, High is True
8	GRND	COMMON	SMCC COMMON	
9	MAI3/	INPUT	MACHINE INPUT 3	5V TTL, High is True
10	GRND	COMMON	SMCC COMMON	
11	MAI2/	INPUT	MACHINE INPUT 2	5V TTL, High is True
12	GRND	COMMON	SMCC COMMON	
13	MAI1/	INPUT	MACHINE INPUT 1	5V TTL, High is True
14	GRND	COMMON	SMCC COMMON	
15	MAO5/	OUTPUT	MACHINE OUTPUT 5	Open Collctr.,
16	GRND	COMMON	SMCC COMMON	Low is true
17	MAO4	OUTPUT	MACHINE OUTPUT 4	Open Collctr,
18	GRND	COMMON	SMCC COMMON	Low is true
19	MAO3/	OUTPUT	MACHINE OUTPUT 3	Open Collctr,
20	GRND	COMMON	SMCC COMMON	Low is true
21	MAO2/	OUTPUT	MACHINE OUTPUT 2	Open Collctr,
22	GRND	COMMON	SMCC COMMON	Low is true
23	MAO1/	OUTPUT	MACHINE OUTPUT 1	Open Collctr,
24	GRND	COMMON	SMCC COMMON	Low is true
25	PL5V	OUTPUT	+5V POWER	
26	GRND	COMMON	SMCC COMMON	

Appendix G

Summary of SMCC Commands

Notations:

R:	Possible Range of parameter
D:	Default parameter
U:	Unit of Quantity
():	Optional
[]:	Equivalent command for Y axis
<CNTRL>	Control Key

G.1 DIRECT COMMANDS

G.1.1 Card Address Select Command

An n = 0 - 9, or a - f for card # 0 - 15.
AA for all cards active simultaneously.

G.1.2 Direct Motion Commands

Q	Quit; stop running the program, stop at end of next block.
R	Run; start executing blocks continuously.
S	Step; execute one block and then stop.
H	Feed Hold (Stop immediately).
=	Return to Pre-Jog Position
h	HOME; Start homing cycle
J(X)[JY]	Jog Positive.
j(X)[jY]	Jog Negative.

G.1.3 System Control Commands

Z	Zero Set (set current position to zero)
k	Kill Servo, stop holding position; allow motor to free-wheel..
q	Abort current move "Step" or "Run" will execute the next move.
\$	Soft Reset.
\$\$	System Reset.
%nn,mm	Feed Rate % Override.
Bnn	Goto start of program if nn is omitted or go to subroutine nn.
PC=v	Set program counter, execution will start at line V.
I	Learn and Store X and Y Positions if buffer is empty, Learn and Replace current position if buffer is full.
znn	Purge Buffer and labels and reserve nn steps for program buffer. Or, purge program only and keep labels if n is omitted.

s	Save Program and Parameters in EAROM
innv	Set parameter inn to value v.
Pn=v	Set Parameter n to value v.
Cmm=A	Define Cmm to be address A. mm is 0 to 64.
Mnn=Cmm.b	Define M function bit at address Cmm at bit position b, b is 0 to 7, nn - is 0 to 64 and mm - is 0 to 31.
Mnn=1/0	Set (turns on) or Reset (turns off) the specified M function bit.
O(X)nn[OYnn]	Open-loop Control, output fixed voltage value to the amplifier. nn is 0, +/- 255 for 0 to +/- 10 VDC.

G.1.4 Status Query Commands

inn	Send value of Stored Initialization/Setup Parameter nn.
innH	Send the maximum allowable value of parameter inn.
innL	Send the minimum allowable value of parameter inn.
innD	Send the factory-set default value for parameter inn.
%	Send the value of the % command.
ix	Send value of all Initialization/Setup Parameters.
f	Send Following Error
p	Send Current Position
v	Send current Velocity
?	Send Status (See Appendix B for Code Description)
Bnn	Send Step or program line number for Label nn, or size of program buffer if nn is omitted.
Bx	Send step or program line number for all labels, 0 to 63.
Cmm	Send Defined Memory Location Address for Cmm.
Cx	Send defined memory location for all addresses defined by all Cmm's 0 to 63.
Pnn	Send Parameter value.
Px	Send value of all parameters 1 to 15.
Mnn	Send defined Mnn parameter
Mx	Send all Mnn parameters
PC	Send value of next program line to be executed.
PB	Send value of Program Buffer size.

G.1.5 Editing Commands

LIST n, m	List m lines of buffer program starting at line n.
DELETE n,m	Delete m lines of buffer program starting at line n.
INSERT n	Insert Additional lines of program starting at line n.

REPLACE n	Overwrite program line n.
MAP n	Reallocate n steps for main program area.

G.1.6 Advanced Command Set

rA, v	Read Memory Data at address A, and send V bytes of data. V is 1 to 7.
dA, v	Read data from address A and display with format.
wA,d	Write Data d to Memory address A.

G.1.7 Control Key Commands

<CNTRL B>	Send binary position commands.
<CNTRL C>	Request binary position Data.
<CNTRL D>	Request binary Status Data.
<CNTRL E>	Request binary P1 and P2 values.
<CNTRL V>	Freeze position command.
<CNTRL M>	(Carriage Return <CR>) causes the SMCC to act on the non-control characters it has received.
<CNTRL H>	Erases the most-recently entered character.
<CNTRL R>	Sets the value of the "%" to that in the i07 parameter.

G.2 BUFFER INSTRUCTIONS

G.2.1 Program Flow Control Commands

STOP	Stop the program.
END	End of Program.
START	Start continuous Motion, execute blocks continuously, without stopping.
RETURN	Return from Subroutine.
FOR n	Start Loop and execute the loop n times.
NEXT	End of Loop.
CASE	Create jump table.
Lnn	Label, nn is 0 to 63.
SET Pn	Set Parameter value.
Innv	Set Initialization Parameter, to value v.
Fnn	Set Feed Rate in encoder counts/second.

Tnn	Move time in 1/1024 seconds.
tnn	Set Accel/Decel Time.
LINEAR	Define Linear Motion.
CIRCLE n	Define CW/CCW Circular Motion.
DIVIDE n	Divide next move into n steps.
X=nn[Y=nn]	Set current X(Y) position to nn.
R(X)[RY]=nn	Set radius in Circular Motion.
GOTO(+/-)nn Condition	Jump a specified number of program lines forward or backward, from the current program line. nn is +/- 63.
GOTO nn Condition	GOTO specified subroutine. do not return.
GOSUB nn Condition	Jump to specified routine and return.

G.2.2 Motion Control Commands

HOME	Go to home position.
Xnn	Move X[Y] axis to n encoder counters (Absolute Move).
AXnn[AYnn]	Move X[Y] axis to the point which is n counts from Home or initial zero position.
Unn[Vnn]	Move X axis n encoder counts from the current position. (Incremental Move)
W(X)nn[WYnn]	Wait time for each axis, equivalent to a move time.
DWELL nn	Stop and hold position for nn/1024 seconds.
POT nn	Controls feed rate, same as % (percent) command.
SLEW nn	Controls the rate that the "POT" command changes the feedrate.
SET Pnn=vvv	Define Variable Pnn to vvv.
SET Pnn=HX	
SET Pnn=HY	
SET Pnn=AX	
SET Pnn=AY	These instructions set Pnn to Home captured position or to the ending position of the current move.
RESET Mnn	Turn off M function.
SET Mnn	Turn on M function.
RESETP	Reset All P variables to Zero.
CASE nn, vvv	Read value of Cnn address and jump depending on vvv.
SET Cnn=aa	Define Memory Address aa as Cnn.
SET Mnn=Cmm.b	Define M function variable Mnn as b-bit of Cmm.
POI nn	Controls feed rate, same as % (percent) command.
SET nn	Set the bit defined by Mnn.
RESET nn	Reset the bit defined by Mnn.
SLEW nn	Controls the rate at which the "POT" command will change the feedrate.
:d	Define parabolic move with ending velocity d.

WUT	Wait until trigger.
^d	Go until Interrupt.
tt d	Triggered move calculation time
q...	Altered Destination move in MDI mode.
DLY	Execute a delay time.
TORQUE	Put SMCC in torque-limited mode.

G.2.3 Advanced Instruction Set

mAA,d	Write data d to Memory location A.
mmAA,d	Same as mAA, d command but executes immediately.
nAA,d	Logical AND Memory Data.
uAA,d	Logical OR Memory Data.
xAA,d	Logical XOR (Exclusive OR) Memory Data.

G.3 PLC Commands/Instructions

IF Condition	Start of Condition.
THEN Action	Do this if condition true.
ELSE Action	Do this if condition not true.
AND Cond./Act.	AND previous Condition/Action.
OR Cond./Act.	OR previous Condition/Action.
END	End of PLC Program.
PLC	List PLC Program.
PURGE	Purge PLC Program.

SUMMARY OF PARABOLIC SMCC i-PARAMETERS
For Brush Type DC Motor Controllers (If i13 = 0)

Notation:

EC Encoder Counts
FR Ratio or Fractional Unit
[] Equivalent commands for Y Axis

i#	FUNCTION	RANGE	UNIT	DEFAULT
i00	Following Error Limit (FAULT)	+/-32767	EC	2000
i01	In-Position Band	0-65535	FR	100
i02	MDI, Manual Data Input Mode	0,1	None	0
i03	Hand Shake Enable	0,1	None	1
i04	Home Search Feed Rate	0 - 255	FR	0
i05	Servo Time	0 - 65535	mSec	487
i06	Reference Max. Feed Rate	0 - 8388607	EC	2000
i07	Time Scale	0 - 65535	FR	65498
i08	Accel./Decel. Time	0 - 65535	≈mSec	512
i09	Position Integration Mode	0,1	None	1
i10	Following Error Limit	0 - 65535	EC	1000
i11	"Feed Hold" Slew Control	0,1	None	0
i12	Move Delay or Calculation Time	0 - 255	mSec	40
i13	MUST BE 0 FOR DC MOTORS	0, 1	None	0
i14	Processor A/D Converter Mode	0 - 3	None	0
i15	Y Axis Disable in 2 Axis Mode	0,1	None	1
i16	Handwheel Operating Mode	0,1,2	None	0
i17	Velocity Display Time Base	0 - 2 ³¹	mSec	15000
i18	Display Mode	0 - 3	None	3
i19	Highest Card Address	0 - f	None	0
i20[i40]	Proportional Gain Constant	0 - 65535	FR	50
i21[i41]	Differential Gain Constant	0 - 255	FR	240
i22[i42]	Velocity Feedforward Gain	0 - 255	FR	10
i23[i43]	Integral Gain Constant	0 - 255	FR	0
i24[i44]	Home Direction	0,1	None	0
i25[i45]	Home Offset	+/-32767	EC	500
i26[i46]	Home & User Flag Controls	0 - 7	None	0
i27[i47]	Hand Wheel Scale Factor	0 - 65535	FR	4096
i28[i48]	Hand Wheel Mode Control	0 - 3	None	3

i#	FUNCTION	RANGE	UNIT	DEFAULT
i29[i19]	Hand Wheel Encoder Control Bit	0 - 19[0-11]	None	3
i30[i50]	Acceleration Feedforward	0 - 255	FR	0
i31[i51]	Jog Feed Rate	0 - 255	FR	200
i32[i52]	Normal PWM Limit	0 - 255	FR	128
i33[i53]	Protective PWM Limit (I2t)	0 - 255	FR	64
i34[i54]	Anti-Backlash Overtravel	+/-32768	EC	0
i35[i55]	Positive Position Limit	-1 to 2^{27}	EC	-1
i36[i56]	Negative Position Limit	-1 to 2^{27}	EC	-1
i37[i57]	Position Range	+/- 2^{27}	EC	0
i38[i58]	Display Scale Factor/Format	n: m.d	FR	1:1.0
i39[i59]	Position Encoder Control Bits	0 - 7	None	3
i60[i61]	Dead Band	0 - 255	FR	0
i62	PLC enable	0,1	None	0
i63	DMA Communication	0,1	None	0
i64[i65]	Reserved	0	None	0
i66[i67]	Reserved	0	None	0
i68[i69]	Reserved	0	None	0
i70[i71]	Reserved	0	None	0
i72[i73]	Reserved	0	None	0
i74[i75]	Reserved	0	None	0
i76[i77]	Reserved	0	None	0
i78[i79]	Range Mode Control	0,1	None	0
i80	Backlash Take up	0 - 15	None	0
i81	Circle Error Tolerance	0 - 15	EC	0

SUMMARY OF PARABOLIC SMCC i-PARAMETERS
For Brushless AC Motor Controllers (If i13 = 1)

Notation:

EC Encoder Counts
FR Ratio or Fractional Unit
[] Equivalent commands for Y Axis

i#	FUNCTION	RANGE	UNIT	DEFAULT
i00	Following Error Limit (FAULT)	+/-32767	EC	0
i01	In-Position Band	0 - 65535	EC	100
i02	MDI, Manual Data Input Mode	0,1	None	0
i03	Hand Shake Enable	0,1	None	1
i04	Home Search Feed Rate	0 - 255	FR	100
i05	Servo Time	0 - 65535	mSec	487
i06	Reference Max. Feed Rate	0 - 8388607	EC	100000
i07	Time Scale	0 - 65535	FR	65498
i08	Accel./Decel. Time	0 - 65535	≈mSec	100
i09	Position Integration Mode	0,1	None	1
i10	Following Error Limit	0 - 65535	EC	1000
i11	"Feed Hold" Slew Control	0,1	None	1
i12	Move Delay or Calculation Time	0 - 255	mSec	40
i13	MUST BE 1 FOR BRUSHLESS MOTORS	0,1	None	1
i14	Processor A/D Converter Mode	0 - 3	None	0
i15	Y Axis Disable in 2 Axis Mode	0,1	None	0
i16	Handwheel Operating Mode	0,1,2	None	0
i17	Velocity Display Time Base	0 - 2 ³¹	mSec	14648
i18	Display Mode	0 - 3	None	3
i19	Highest Card Address	0 - f	None	0
i20	Proportional Gain Constant	0 - 65535	FR	1000
i21	Differential Gain Constant	0 - 255	FR	10
i22	Velocity Feedforward Gain	0 - 255	FR	10
i23	Integral Gain Constant	0 - 255	FR	0
i24	Home Direction	0,1	None	0
i25	Home Offset	+/-32767	EC	0
i26	Home & User Flag Controls	0 - 7	None	5
i27	Hand Wheel Scale Factor	0 - 65535	FR	4096

i#	FUNCTION	RANGE	UNIT	DEFAULT
i28	Hand Wheel Mode Control	0 - 3	None	3
i29	Hand Wheel Encoder Control Bit	0 -19[0-11]	None	3
i30	Acceleration Feedforward	0 - 255	FR	0
i31	Jog Feed Rate	0 - 255	FR	128
i32	Normal PWM Limit	0 - 255	FR	255
i33	Protective PWM Limit (I2t)	0 - 255	FR	128
i34	Anti-Backlash Overtravel	+/-32768	EC	0
i35	Positive Position Limit	-1 to 2 ²⁷	EC	-1
i36	Negative Position Limit	-1 to 2 ²⁷	EC	-1
i37	Position Range	+/- 2 ²⁷	EC	0
i38	Display Scale Factor/Format	n: m.d	FR	1:1.0
i39	Position Encoder Control Bits	0 - 7	None	3
i40	Encoder Counts / Electrical Revolution	0 - 65535	EC	2048
i41	Phase Advance Scale Factor	0 - 255	None	12
i42	Induction Motor Magnetization Current	0 - 255	None	0
i43	Commutation Method	129 - 131	None	130
i44	Number of Motor Phases	0, 1	None	1
i45	Induction Motor Slip Gain	+/-32767	None	0
i46	Y-User Flag	0, 1	None	0
i47	N/A	4096	None	4096
i48	Star / Delta Connection	0, 1	None	0
i49	N/A	3	None	3
i50	Triple-Cycle Feedback	0, 1	None	0
i51	N/A	200	None	200
i52	Motor Phase Finding Time	0 - 255	mS x 43	40
i53	Phasing Current Value	0 - 255	None	10
i54	Hall Effect of Resolver Offset	+/-32767	EC	0
i55	N/A	-1	None	-1
i56	N/A	-1	None	-1
i57	N/A	0	None	0
i58	N/A	1 : 1.0	None	1 : 1.0
i59	N/A	3	None	3

i#	FUNCTION	RANGE	UNIT	DEFAULT
i60[i61]	Dead Band	0 - 255	FR	0
i62	PLC enable	0,1	None	0
i63	DMA Communication	0, 1	None	0
i64[i65]	Reserved	65464	None	65464
i66[i67]	Reserved	0	None	0
i68[i69]	Reserved	0	None	0
i70[i71]	Reserved	0	None	0
i72[i73]	Reserved	0	None	0
i74[i75]	Reserved	0	None	0
i76[i77]	Reserved	0	None	0
i78[i79]	Range Mode Control	0, 1	None	0

INDEX

PAGE

1/T Encoder period		8-8, A-11
A0		3-1
AA		3-5
Acceleration Feedforward Gain		3-11, A-17
Acceleration/Deceleration Time		A-4
Address Select Command		3-5
Altered Destination Move		5-15
Analog Command Input, Processor A/D Converter		3-13, A-7
Analog Input Output		2 3
Backlash Compensation		3-12, A-18
Backlash Take Up		A-27
Buffer Instructions		5-1
Card Address DIP Switch		3-2
Card Address Select Command		3-5
Circle4 Mode		8-2
Circle Error Tolerance		A-27
cm, Count per Revolution		3-7
Control Character Commands		4-11
Controlled Hold Mode (Feed Hold Slew Control)		4-22
Command		
\$	Soft Reset	4-2
\$\$	System Reset	4-2, 4-20, G-1
%nn, mm	Feed Rate Override	4-2, 5-12
=	Return to Pre-Jog	4-1
?	Send status and error codes	4-9
An	Address Select	3-5
AND	AND Condition/Action	6-1
AXnn [AYnn]	Absolute Move	5-13
bnn	Go to Beginning	4-3
Rnn	Send Map	4-10
Bx	Send Map	4-10
CASE	Create jump table	5-7
CIRCLE N	Define circular motion	5-10
Cmm	Send address definition	4-10
Cmm = A	Defines address	4-4
CNTRL	Control Character Command V M H	4-11
Cx	Send all address definition	4-10
dA, v	Read data	4-5
DELETE n, m	Delete program steps	4-6
DIVIDE n	Divide motion blocks	5-12
DLY n	Execute a delay time	5-16
DWELL nn	Wait	5-13
ELSE	Do if condition not true	6-1
END	End of PLC program	6-1
END	End of program	4-1
f	Send following error	4-8
Fnn	Set Feed Rate	5-9

INDEX

PAGE

FOR n	Start Loop	5-2
GOSUB nn	Subroutine Call	5-5
GOTO (+/-) nn	Relative Branch	5-4
GOTO nn	Subroutine Branch	5-5
H	Feed Hold	4-1
h	Home	4-1
HOME	Go to Home Position	4-1, 5-13
i00 - i81	i-parameters	Appendix A
IF	Start of Condition	5-5, 6-1
innD	Send Default Value	4-4, 4-8
innH	Send Maximum allowable value	4-8
innL	Send Minimum allowable value	4-8
innv	Set Initialization Parameter	4-4, 4-8
INSERT n	Insert Program Line	4-6
ix	Send all inn	4-8
j(X) [jY]	Jog Negative	4-1
J(X) [JY]	Jog Positive	4-1
k	Kill Servo	4-2
l	Learn	4-3
LINEAR	Define Linear Motion	5-9
LIST n, m	List Program Lines	4-6
Lnn	Label	4-1
mAA, d	Write Data	5-9
MAP n	Reallocate	4-6
mmAA, d	Write Data	5-9
Mnn	Send Mnn Definition	4-10
Mnn = Cmm.b	Defines Bit	4-5
Mx	Send all Mnn Definition	4-10
nAA, d	Logical AND	5-9
Next	End of Loop	5-4
O (x) nn [OYnn]	Open Loop Output	4-2
OR	OR Condition / Action	6-1
p	Send Position	4-8
PLC	List PLC Program	6-1
Pnn	Send Parameter Value	4-4, 4-10
Pnn = v	Define Variable	4-4
POT nn	Feed Rate Override	5-12
PURGE	Purge PLC Program	6-1
Px	Send Parameter Value	4-10
q	Abort	4-1
Q	quit	4-2
q . . .	Altered Destination Move	5-15
R	Run	4-1
R = nn	Set X and Y radii	5-10
R (X) [RY] = nn	Set Radius	5-10
rA, v	Read Memory Data	4-5
REPLACE n	Overwrite	4-6

INDEX

PAGE

RESET nn	Turn Off M Function	5-6
RESETP	Reset All P to Zero	5-2, G-4
RETURN	Return from Subroutine	5-1
s	Save Program	4-4
S	Step	4-1
SET Cnn = aa	Define Memory Address	5-7
SET Mnn = Cmm.b	Define Address Location	5-7
SET nn	Turn on M Function	5-6
SET Pnn	Set Parameter Value	5-1
SLEW nn	Controls Slew Rate	5-12
START	Start Continuous Motion	5-1
STOP	Stop the Program	5-1
THEN	Do This if Condition True	6-1
TORQUE	Torque Limited Mode	5-9
tnn	Set Accel / Decel Time	5-9
Tnn	Time Mode	5-9
tt n	Triggered Move Calculation Time	5-15
uAA, d	Logical OR	5-9
Unn [Vnn]	Relative Move	5-13
v	Send Velocity	4-8
w (X) [WY]	Wait	5-13
wA, d	Write Data	4-5
WUT	Wait Until Trigger	5-13
X = nn [Y = nn]	Position Set	5-9
xAA, d	Logical XOR	5-9
Xnn[Ynn]	Absolute Move	5-13
y	Data Gathering	8-8
z	Zero Set	4-3
znn	Purge Buffer	4-4
^d	Go Until Interrupt	5-15
:d	Define Ending Velocity	5-13
Command Format		3-4
Communication – Parallel		3-2
Communication DIP Switch		3-2, C-1
Connectors and PIN Assignments		Appendix F
Count Per Revolution		3-7
CTS, RTS		3-3
Data Gathering		8-8
DB-25 Connector Pin		
Deadband		3-13
Diagnostics – Power up, communication		3-3
Differential Gain Constant		3-11, A-11
Digital Outputs		2-3
Dip Switch		C-1
Direct Commands		4-1
Display Type		A-9
DMA Communication		8-10

INDEX

PAGE

DMA, Host to Parabolic SMCC	8-10
DMA, Binary Parabolic SMCC, to Host	8-11
Mode Control	A-16
DSR, DTR	3-4
Duplex, HALF, FULL	3-4
E2, E3	3-1, 3-3, E-1
E7, E8	3-1, E-1
E Points, E1 - E50	E-1
External Pulse (Handwheel) Input	3-13
External RAM Address Definition	D-10
Fault / Error Code	B-1
Feed Hold Slew Control	A-5
Feed Pot Mode	A-7
Following Error Limit	A-1, A-5
Go Until Interrupt	5-15
Hands-On Command Example	4-12
Acceleration / Deceleration	4-16
Card Selection and Basic Move	4-13
Dual Axis Control	4-18, 5-21
Dual Axis Coordinated Motion	5-19
Following Error	4-15
Jog Speed	4-15
Machine Input Output Functions	5-23
Resetting SMCC	4-20
Subroutine	5-18
Time Specified Move	5-17
Unit of Position	4-13
Usage of Buffer Instructions	4-19, 5-16
Elliptic Motion	5-22
Multiple SMCC Coordinated Motion	5-22
Handshake Enable	3-3
Handshake Enable in Host Communication	A-2
Handwheel Encoder Control Bits	A-16
Handwheel Mode Control	A-16
Handwheel Operating Mode	A-8
Handwheel Scale Factor	A-14
Highest Card Address	A-10
Home Direction	A-12
Home Feedrate	A-3
Home Flag and User Flag Control Bits	A-13
Home Offset	A-13
Host Communication – Set Up	3-1
In-Position Band	A-2
Instruction Format	3-4
Integral Gain Constant	A-12
Internal RAM Address Definition	D-3
J1 - J12 Connectors	F-1

INDEX

PAGE

JMACH, J1 Connector	F-4
Jog Feedrate	A-17
Jog Speed Control Mode	A-7
JPAR-RS232, J2	F-7
LED Display Codes	B-1
M-Function I/O	4-11
Machine Input Address	D-1, D-2
Machine Input, Output	4-11
Machine Output Address	D-1, D-2
Manual Control Inputs	2-4
Manual Data Input Mode	A-2
Memory and I/O Map	Appendix D
Memory Capacity	2-1
Motion Commands	4-1
Motor Type Selection	3-6
Motor / Amplifier Connection	3-1
Move Calculation Time	A-6
Normal Data Processing Mode	3-9
Null Modem	3-1
On-Board Dip Switch	C-1
Operations	3-16
P Variable Arithmetic	5-3
Panel Interface Address	D-1
Parabolic Profiling	8-1
CIRCLE4	8-1
Direct	8-1
Point-to-Point S-Curve	8-4
S-Curve	8-2
PID Controller	A-10
PLC Action Format	6-3
PLC Condition Format	6-3
PLC Direct Commands, Buffer Instructions	6-1
PLC Enable	3-15, A-25
Point-to-Point "S" Curve Move	8-4
Position and Display Scale and Format	A-19
Position Encoder Mode Control Bits	A-20
Position Integration	3-11, A-5
Position Limit, Range	A-18, A-19
Position Integration Mode	A-5
Position Offset Mode	A-13
Processor A/D Converter Mode	A-7
Program Control Command	4-2
Program Definition / Control Instructions	5-1
Program Editing Commands	4-6
Proportional Gain	3-11, A-10
PWM Limit Control	A-17
Range Mode Control	A-26

INDEX

PAGE

Rapid Hold Mode	4-22
Reference (Maximum) Feedrate	A-3
Reset SMCC	4-20
RS-232 Port	2-1
RTS, CTS	3-3, E-2
RXD, TXD	3-3, E-2
“S” Curve	8-2
Servo Performance	2-1
Servo Time Period	A-3
Set Up Encoder	3-7
Set Up Home	3-15
Set Up Initialization Command	3-6
Set Up Limit	3-3-8
Set Up Position/Velocity Display	3-14
Set Up System Reference, i06, i08, i31[i51]	3-7
Shortest Distance	A-26
SMCC	1-1
Software Positive/Negative Limit	A-18
Specification – SMCC	2-1
Status Byte	4-9, B-1
Status Codes	B-1
Status Query Commands	4-8
Summary of i-Parameters	G-6
Summary of SMCC Commands	G-1
SYNC Clock	3-1
Test Points, TP1 - TP11	E-7
Three Point Smoother	8-2
Time Move	5-8
Time Scale	A-4
Timer Usage (PLC)	6-4
Tuning	3-10
TXD, RXD	3-3, E-2
User Flag	A-13
Velocity Control	2-2
Velocity Control Mode	A-7
Velocity Feedforward	A-12
Version Number Address	D-1
Wait Until Trigger	5-13
Y Axis Disable	A-8

BALDOR[®]
MOTORS AND DRIVES

BALDOR ELECTRIC COMPANY

P. O. Box 2400

Fort Smith, AR 72902-2400

(501) 646-4711

FAX (501) 648-5792

© Baldor Electric Company
MN1209

Printed in USA
11/95 C&J 5000