BALDOR®

MOTION PRODUCTS GROUP

Smart Motion Control Card

(Parabolic Version)

Operations Manual

Version 2.01

3-15-91

5205 HIGHWAY 169 N * PLYMOUTH,MN 55442 * (612 557-9250)

TM-420-B-1

TABLE OF CONTENTS

Chapter	Title	Page
1	Overview	
2	Specification	
3	Getting started	
3.1	Setting up the hardware	3.1
3.1.1	Motor/Amplifier connection	3.1
3.1.2	Setting up for Host communication	3.1
3.1.3	Power-up and Communication diagnostics	3.2
3.2	Communication and Command format	3.4
3.2.1	Command/Instruction format	3.4
3.2.2	Card Address Select command	3.5
3.3	Set up Initialization parameters	3.5
3.3.1	Motor type selection	3.6
3.3.2	Encoder Setup	3.6
3.3.3	System Reference Speed Setup	3.6
3.3.4	System Limit Setup	3.7
3.3.5	Parameter For Special Application	3.8
3.3.6	Operational Diagnostics and System Tuning	3.8
3.3.7	Use of External Pulse (Handwheel) input	3.11
3.3.8	Use of Analog Command Input	3.12
3.3.9	Position/Velocity Display Setup	3.12
3.3.10	Home Setup	3.13
3.3.11	PLC Enable/Disable	3.14
3.4	Simple Operations	3.14
4	Direct Commands	
4.1	Motion Commands	4.1
4.2	Program Control Commands	4.2
4.3	Program Editing Command	4.5
4.4	Status Query Commands	4.7
4.5	Control Character Commands	4.9
4.6	Hands-on Command examples	4.10

Chapter	Title	Page
5	Buffer Instructions	
5.1	Program Definition/Control Instructions.....	5.1
5.2	Motion Definition Instructions.....	5.6
5.3	Motion Instructions.....	5.9
5.4	Hans-on Application programs.....	5.13
6	PLC Programming	
6.1	Direct PLC Commands.....	6.1
6.2	PLC Buffer Instructions.....	6.1
6.3	PLC Program examples.....	6.5
7	Application Programs	
	Example 7.1 Single Axis Torch Cutter.....	7.1
	Example 7.2 Two DC motors in Pick and Place Operation	7.2
	Example 7.3 Serpentine Pattern.....	7.4
	Example 7.4 Thumbwheel Switch Application.....	7.5
	Example 7.5 Glue Dispenser program.....	7.10
	Example 7.6 Altered Destination On Trigger.....	7.12
	Example 7.7 Multiple SMCC Sequencing Method.....	7.13
	Example 7.8 Automated Encoder line Counting program..	7.16
	Example 7.9 "S" curve Accel and Decel.....	7.17
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 Circle4) Mode.....	8.2
8.1.3	General Algorithm for Point-To-Point S-Curve Moves...	8.4
8.2	1/T Encoder Period.....	8.7
8.3	Data Gathering.....	8.7
8.4	DMA Communications.....	8.9

Appendices

A	Glossary of Initialization/Setup Commands
B	SMCC Status Codes
C	On-Board DIP Switch
D	SMCC Memory and I/O Map
E	SMCC E-Points and Test Points
F	SMCC Connectors and PIN Assignments
G	Options and Accessories
H	Summary of SMCC Commands

Index

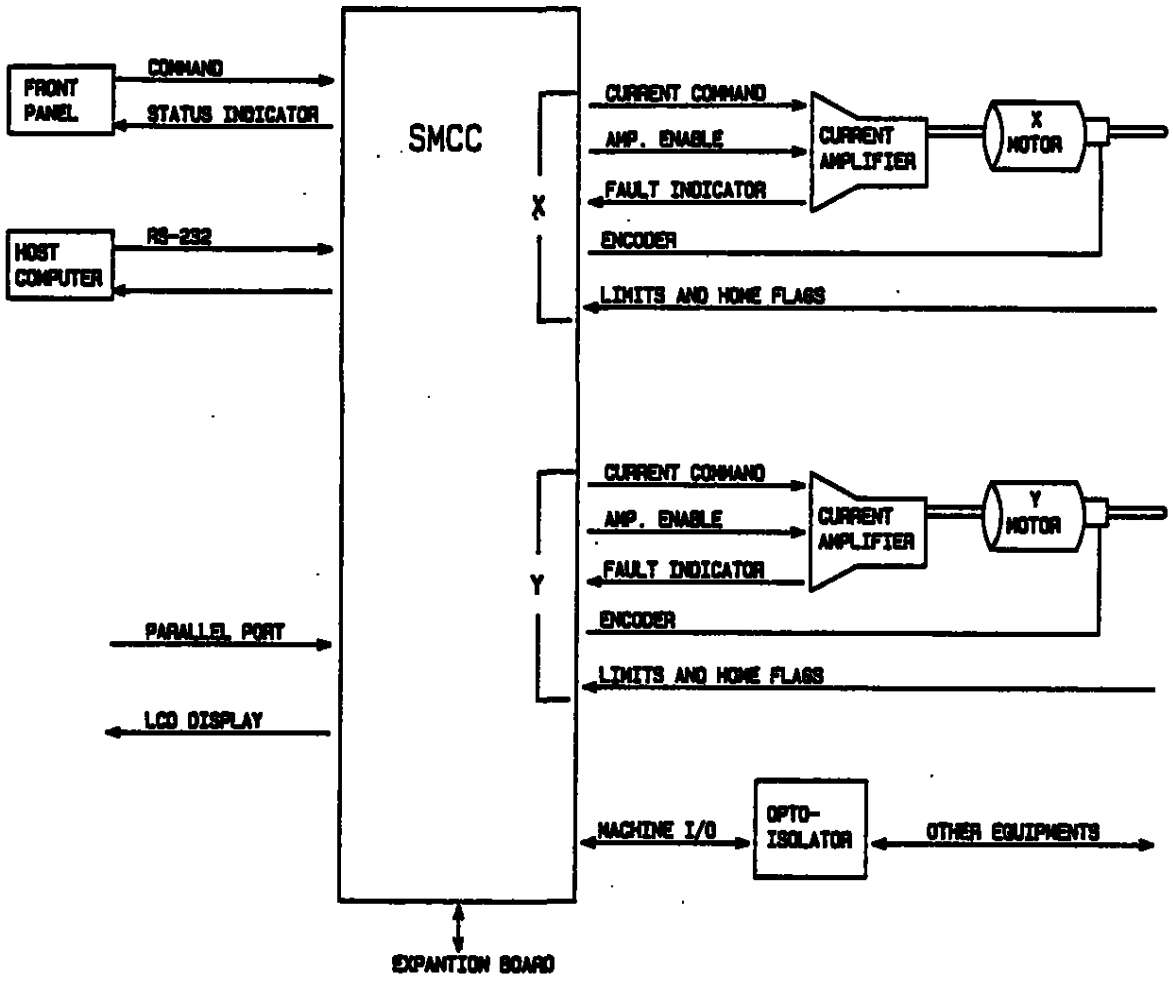
1. OVERVIEW

GENERAL:

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

FEATURES:

* Fast 16-bit pipelined custom microprocessor control.
* Compact size, fits in amplifier chassis or in stand-alone enclosure.
* Host communication via RS-232 serial port, or high speed bi-directional 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 amplifier 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 amplifiers.



TYPICAL APPLICATION OF SMCC

2. SPECIFICATION

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 amplifier command output (Usually provided by the amplifier). Current requirements are 50mA on positive supply and 15mA on negative supply.
* +/-15VDC, 100 mA is required if optional D/A, A/D, or R/D converter are installed.

2.3 RS-232 PORT

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

2.4 PARALLEL PORT

* An alternative host communication port: 8-bit parallel data transfer, with handshake; Average per character transfer time is up to 120 microseconds; 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 (upto 16 channels without external decoding) or used as 8 additional TTL Outputs.

2.5 MEMORY CAPACITY

* 8K byte RAM and 8K byte EEPROM standard; can store 1790 steps of program.
* 32K byte batterry-backed RAM optional; can store 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 RPM range dependent on encoder 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 amplifier), amplifier disable and amplifier fault input signals.
* All signals are referenced to the amplifier 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 +/-10V DC (torque or velocity) error outputs and 9 bits resolution.
* Software-selectable sinusoidal, trapezoidal, or six-step phase commutation waveforms for brushless DC and AC induction motors.

2.13 ANALOG INPUTS

* 4 analog inputs:
  - 2 are used to monitor motor current and provide programmable i2t protection
  - 1 used as 0 to 100% Feed rate control with simple potentiometer input
  - 1 used as bipolar velocity control with range multiplication using jog inputs.

2.14 DIGITAL OUTPUTS

* +/-limits for X and Y axis
* Amplifier disable output
* Amplifier 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 amplifiers equipped with error code generation, 3 of the 7 inputs may be used to display amplifier error codes. All inputs may be read under program or PLC control.
* Five general purpose Output Channels: Open collector, 30V, one at 60mA (MAO1), four at 200mA (MAO2-MAO5). All outputs are programmable "on the fly" by program or PLC control .

2.15 MANUAL CONTROL INPUTS (REQUIRES CONTROL PANEL)

* The SMCC supports following dedicated Manual direct inputs. (Control Panel with momentary toggle switches.)

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.
PREJ/	=	Return to pre-jog or pre-handwheel position.
STRT/	R	Starts continous program execution.
STEP/	S	Executes one bolck of data at a time.
STOP/	q	Stops running immediately, resets program.
HOME/	h	Starts X and Y axes on a Homeing cycle.
HOLD/	H	Stops motion immediately.

* Control Panel Outputs:

Name	Function
IPLD/	In-Position indicator.
BFLD/	Buffer Full indicator.
ERLD/	Amplifier Fault indicator.
EPLD/	End pf progra indicator.
ITLD/	Current limit indicator.
FILD/	Following error 1.
F2LD/	Following error 2.

3. GETTING STARTED

3.1 SETTING UP THE HARDWARE

3.1.1 MOTOR/AMPLIFIER CONNECTIONS

All wiring should properly be made according to the chassis manual. As a minimum, connection to encoders, amplifiers and motors, and RS-232 communication link to host computer should be done.

Wiring of limit switches must be done. If limit switches are not used, disable limit protection by grounding 4 limit switch lines XPLL, XMIL, YPLL, YMIL. Consult chassis manual for the exact location of these pins.

Make sure that all 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

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 Card address is set as A0, it must always act as a "master" supplying SYNC clock to other cards (Default).
(Jumper on E2 installed, Jumper on E3 removed).
(e) Since a direct link between Host computer and SMCC is used normally, SMCC is configured as a "Null-Modem" so that a standard 9 line Modem Cable fits without modifications. (Jumpers on 1-2 and 3-4 on both E7 and E8).

If a single SMCC is to be used without daisy-chaining, all default configuration should work properly. Connect RS-232 cable between SMCC and the host computer, and run a communication program (such as Crosstalk, Procom, etc). You may obtain "Monitor" communication program from Baldor free of charge. It operates on PC or compatible computer, and has many convenient features for host communication. Host computer line parameters must be set as 8-bit data, 1 stop bit, no parity, half duplex with Baud rate the same as set by the SMCC DIP switch.

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 38400 Baud
1	1	1	Serial 19200 Baud

If parallel communication is to be used, you may first setup with RS-232 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 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 SMCC's. Install jumpers E2 and E3 as follows.

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

For daisy-chain connection, use factory supplied cable or follow instructions on chassis manual.

3.1.3 Power-up and Communication Diagnostics

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

STEP 1. Turn on SMCC power supply ON while amplifier bus power is remained OFF. Observe the on-board 7-segment LED display. It may display one of the following patterns.

(1) ".O" (left dot), SMCC is "in-position" and no errors are found. Go to STEP 2.

(2) "O." (right dot), following error. proceed ahead to set up correct parameters. Go to STEP 2.

(3) ".O." (two dots), 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 factory or distributor. Otherwise, check +5V supply is in between 4.75 and 5.25 volts.

(4) LED OFF ; No power on SMCC. Turn off the power, check power supply connections and blown fuse and try STEP 1 again.

STEP 2. Run communication program from the host computer, and type "U" and Carriage Return (CR). Since you are in half duplex mode, "U" character should be printed on the screen. Cursor on the host computer screen should respond with Line Feed and "?". This is a desired response telling that SMCC is communicating properly. Go to next section. Otherwise, check the following.

(1) Set up as "full duplex". If "U" 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 Baud rate is the same as the rate set by the SMCC DIP switch.

(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, SMCC jumpers E7 & E8 can be changed to adapt to the null modem port. See Appendix E for the description of jumpers.

(4) Host computer port is not setting the "RTS" hand shake line high. SMCC supports hardware handshake lines RTS and CTS. Required interconnect wires between SMCC and host computer when daisy-chaining operation is not used 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.

(5) If communication is erroneous, check the source of noise. Noise may come if cable is too long, or too close from the power line wire etc. Proper shielding, or 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 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 SMCC through RS-232 link, The SMCC stores that command but does not process it until a CR character is received. When a line of command is processed, 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 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 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 (CR)        ;valid command to set maximum Feed rate.
F1024(CR)          ;valid.
 F 1024            ;Blank OK.
```

3. Backspace (Hex 08, or Control-H) 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. 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 address command is detected, commands are stored in a new switched card.

5. Each card can only receive one command line at a time. There are exceptions (e.g. Card Selection Command, Label Command).

6. When Engineering unit is used as a command parameter, 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 SMCC so that 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 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. "AA" command selects all cards simultaneously.

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

An initialization command "inn" without any parameter value is interpreted as parameter value query, where nn could be any number between 00 - 81. The command "ix" will be responded with all parameters.

```
[Example 3.2]

A0          ;Card A0 Enabled.
i00         ;Query to check Following error Fault Limit
            ;SMCC will respond with currently stored values.
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 operation conditions, such as Servo Gains, Following Error bands, Maximum reference speed.. etc. i-parameters are loaded and tested at the factory, follow the instruction to verify factory setting and change if necessary. Appendix A gives detail description for each i-parameter.

There are three ways to change these i-parameters:

1. By Direct Commands "inn".
2. By program using "Inn" instruction.
3. By using "Monitor" program, a dedicated IBM PC program.

In this section, We use direct command inn to set up the Initialization parameters. As a convention, a pair of brackets [ ] means equivalent command for Y axis, so i20[i40] means 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 the interconnections to the amplifier, 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 amplifiers and do not connect motor to the load.

Connects power to the amplifier/motor assembly only when a series of i-parameter has been properly set.

3.3.1 Motor Type Selection

** Set i13 to 0 for DC Motor (brushed) or its equivalent. Certain amplifiers (such as Baldor's BTS-15 Brushless Amplifier) have built-in commutation function and the amplifier and motor acts as a simulated DC motor. In this case, configure it just as the DC motor. For others drive combination, refer to Appendix.A, i13.

** Set i15 to 0 (Two axis) or 1 (X axis only).
Note that even in a one-axis application, the limit switches for Y axis must be grounded, otherwise SMCC will not make any moves.

3.3.2 Encoder Setup

** Set i39[i59], Set Position Encoder Mode Control Bits. First, proper choice must be made in selecting encoder type and 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 we are using Quadrature Decoding and x4 Encoder multiplication factor. Setting i39[i59] to 7 has the same effect except reversed direction. If motor and encoder lines were connected as instructed by the chassis manual, values 0 - 3 should work properly. I39 value of 4 to 7 may be used to reverse the direction of encoder reading (same effect as reversing A and B channel).

SMCC uses count value as the principal means of representing distance or velocity. If mm lines per revolution encoder is attached, and n multiplication factor is selected, Count per revolution "cm" is defined as

```
cm = mm x n    (count per revolution)
```

For example, if mm = 500 lines/rev. and n = 4 (i39 = 3), then cm = 2000. This means whenever motor turns one complete revolution, the distance is recognized as 2000 counts in SMCC. For most of the application, 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 reference

velocity in encoder counts per second. One motor revolution corresponds to the counter value of "cm". For example, if 500-Lines Encoder is configured with x4 multiplication, and wish to set maximum motor speed at 1200 RPM (20 rps), then set i06 to 500 x 4 x 20 = 40,000 [count]

** Set i31[i51] (Range 0 - 255), Jog feedrate (speed), in terms of count per sec. Jog speed is calculated by

Jog Speed = (i31[i51] x i06) / 256

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

** Set i08 (100 - 50000) for Acceleration/Deceleration time roughly in mSec (actually 1/1024 sec.). For example, i08 2000 sets acceleration and/or deceleration time to be about 2 Sec. whenever speed change occurs.

The acceleration and deceleration time must be set to a value the amplifier and motor system can handle. Too small a value causes the amplifier unable to react in the specified time (amplifier over-current limit may occur) or results in too much overshoots in motion 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 following error fault in a frequent speed change application.

i00 allows both positive and negative limit, if you select positive value, an Over-limit condition will cause the SMCC hang up and only reset the whole system can bring it to normal; if you select negative value, an over-limit condition will stop the motions but 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 condition.

** Set i10 1000 (default) for the following error limit in count. When following error exceeds the set value, 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 the range, SMCC turns on in-position indicator.

Once in-position been detected, 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 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. 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 the half of the maximum amplitude. Adjustment may be necessary for proper overload protection.

** Set i33[i53] 64 (default) as a Protective PWM Limit value.
When overload is detected, this limit is effective instead of i32[i52] until 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 -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 Parameter For Special Applications

For most of the operation, You do not need to alter following i-parameters. Verify and make correction 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, Processor sends <LF> character after receiving <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 default time scale. Do not attempt to change i05 and i07. See Appendix A for detailed description of i-parameters.

** Set i14 0 when 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]

```
                ;Make sure that amplifier bus power is still off.
q               ;to enable save, SMCC must be in position.
s               ;Save updated parameters
$$              ;Reset the system
A0              ;Card select
p               ;Obtain current position POS1
                ;Now, turn the shaft one revolution in CW
                ;direction
p               ;Obtain new position POS2. Let DIST = POS2 - POS1.
                ;If DIST is positive, CW is positive direction.
                ;Otherwise, CCW is positive direction.
                ;DIST should be close to the "cm" value calculated
                ;above. If not, check encoder resolution again.

                ;turn off the power.
                ;This time, turn on power with amplifier bus
                ;power on.
                ;Wait until system resets
                ;At this stage, SMCC should hold the shaft.
                ;See NOTE 1 below.
A0              ;Select one card (or any other card number).
O20             ;Try open-loop output command. Motor should turn
                ;in a forward direction as defined above.
k               ;Kill servo, motor will coast to stop.
                ;See NOTE 2 below.
O-20            ;This time, motor should reverse rotation.
Q               ;Quit Open-loop mode
J               ;See if motor jogs, and check jog speed as
                ;calculated by i31[i51] and i06.
                ;Check following error.
j               ;try jog in the other direction
                ;See NOTE 3.
k               ;kill servo. No output to amplifier.
                ;This command can be issued whenever unpredicted
                ;situation occurs.
```

NOTE 1.

(1) Shaft is holding but stiffness is very low. - Needs tuning. Increase i20 value.

(2) Shaft oscillates back and forth - Too much gain, or needs damping. Increase i21. if it doesn't help, decrease i20.

(3) Runaway in one direction - Possibly encoder connection is backward, or amplifier has offset. Check encoder connections and amplifier adjustments.

NOTE 2.

(4) If motor rotates backward, both encoder and motor connections are reversed. Correct

wiring.
(5) If motor does not turn and tries to hold position, check 4 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 (and to be adjusted to a proper value upon observation of the motion). SMCC use proportional gain to calculate and output an error correction signal in the direction proportional to a observed Following Error. As i20[40] increases, the system tends to be stiffer. If the value is too big, the system may oscillate.

** Set i21[i41], differential gain constant. Set to 20 initially (and to be adjusted to different value). 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[43], the integral gain.
Set to 10 initially (and to be adjusted to different value). High integral gain helps reducing the following error, but at the same time, it introduces oscillation to the system, you may keep this gain low for stability reason.

** Set i30 (Acceleration Feedforward gain) to zero (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 user to reduce the following error generated during acceleration and deceleration. However, all others servo gain should be adjusted satisfactorily first and then i31[i50] should be increased gradually until an acceptable following error is achieved.

The "Monitor" Program provides help on tuning servo systems. It utilizes Parabolic SMCC's Data Capturing capability (Chapter 8) to plot both the desired and actual position data, as well as velocity and following error.

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

```
[Example 3.4]

Q     ;Quit from previous kill command
      ;check the stiffness of the system. See if it
      ;oscillates.
J     ;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 motor is connected to the load, the dynamic performance may change significantly. Some refinement on following error limits, acceleration/deceleration time and Servo constants are 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 proper guidelines.

The SMCC has special parameters to compensate the system imperfection:

** 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 specify the number of encoder counts per background cycles during the jog and handwheel moves, for detail refer to Appendix A, i80 and i34[i54].

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

Following steps are optional, set these parameters if necessary. Further description on i-parameter, please refer to Appendix A.

3.3.7 Use of External Pulse (Handwheel) input

```
**  Set i16
        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 (4096).
This parameter determines the ratio of handwheel count vs. motor encoder count, a value of 4096 sets 1:1 relation. (See appendix A, i27)

3.3.8 Use of Analog Command Input

SMCC has provisions to connect +/-10V analog signal inputs to control the speed. It has 4 different modes of operation, set by i14.

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

        i14              Description
         0     if analog command input is not used.
         1     to control feedrate override.
         2     to control jog speed.
         3     for velocity command servo (no position control).
```

3.3.9 Position/Velocity Display Setup

** Set i18 to select Display type according to the following table.

```
        i18              Display Type
         0     Single Line 8 Character LED (For each Axis)
         1     Dual Line 24 Characters/Line LCD (For Each Axis)
         2     Single Line 40 Character LCD (For Each Axis)
         3     Dual Line 40 Characters/Line LCD (For Dual Axis)
```

** Set i38[i58] 1:1.0 (default) for position Display Scaling
i38 [i58] opens the possibility to use user's choice of unit in position representation. 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)
5 pitch lead screw is attached to the motor shaft.
If i38 is set to 10000:100.2, then 1 inch move corresponds to 10,000 counts, and will be displayed 1.00 (inches).

When i38[i58] takes 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
```

```
X3.0              ;Same Instruction when i38 = 10000:100.2
```

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

i17 = (60,000,000) / 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

```
i04 26     ;Homing speed of 10% of Max Speed
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 (0 - +/-32767) for Home offset

** Set i26[i46] to set Home Interrupt control bits according to the following table.

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]

```
i24 1              ;When HOME command is received, SMCC starts
                   ;to turn positive direction until Home Flag
                   ;bit
i26 1              ;to High (by Limit Switch), and Encoder C
                   ;channel
i25100             ;is detected (Rising Edge). At this point,
                   ;SMCC moves 100 counts backward which is Home
                   ;Position.
```

Refer to Appendix E and F for correct pinouts for 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 following commands, the assumption is made here that the SMCC is up and running with both X and Y axis.

[Example 3.12]

```
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 are 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 from SMCC. It will
      ;respond with the 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 position shows a few counts,
      ;such as "4" on an axis, the corresponding Following
      ;error will be a "-4".
J     ;Observe that X axis begins to move in positive
      ;direction.
p     ;Use "p" or "f" to monitor the motion progress.
      ;This demonstrates that You may request from and
      ;send data to SMCC while it is moving.
j     ;Jog in the negative direction
i06   ;SMCC will send the current value of i06, the Max.
      ;reference speed.
i06 20000
      ;20000 is the new reference speed, You may enter any
      ;other value, Enter bigger number will increase speed.
      ;smaller value will decrease the speed.
j     ;Issue another "j" command to stop the jog motion,
      ;Repeat previous jog command will stop Jog motion
      ;Of course, "q" command 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 X axis reverses direction.
```

```
jY    ;Note that Y axis reverse 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 SMCC was before
      ;the jog was started):
=     ;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 Prejog position,
      ;check the velocity of the motor.
i021  ;This places the SMCC in MDI mode
i020  ;This makes SMCC exit the MDI mode.
```

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.) be processed only when the motion is stopped, and in-position.

For easy reference, Direct commands are divided into several groups according to their functions. As a convention, Parentheses ( ) denotes an optional entry; Brackets [ ] indicate an equivalent command for 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 R command, the system must be in-position. So if the system is in "k"(killed) status, First, issue a "q" (abort), and then "R" (Run).

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

H -- Feed Hold (Stop): Stop running program immediately and hold current position. To resume motion, and complete current block, use "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 "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 a toggle Jogging condition.

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

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

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

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

Q -- Quit : Quit a running program at the end of the current block. Also used to terminating Jogging or "O" (Open Loop Output) Command.

k -- Kill Servo: This command sets error output to zero, and opens the servo loop. It is used to disable the amplifier 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: "O" command forces the SMCC sending a torque command with user specified value to its velocity error output. The nn has a range of 0 to +/- 255, where 0 produces zero volts out and 255 produces 10.0 volts out (as a Current Amplifier Command Input). The "O" command automatically disables the servo loop and is very useful for test and calibration of the amplifier velocity loop.

```
[Example 4.1]

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

Following commands are for special applications only

%(nn(,mm)) -- Feed Rate % Override: Range of nn is 0 to 65535, and default is the value of i07, or 65498. The optional entry mm is a slope increment number added to 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 "SLEWmm" commands which can be written separately.

4.2. PROGRAM CONTROL COMMANDS

$ -- Soft Reset: Initiates phase finding operation and preserves all internal data. Usable to correct Phasing change and Motor type etc. only.

$$ -- Reset: It essentially causes a complete reset to occur. But this cannot change Baud rate and Card address. To make a hardware reset, press INIT button at Front Panel.

Z -- Zero Set: This command sets current position to be zero position. It is allowed only when the motion is stopped. The equivalent buffer instructions are X = nn and Y = nn where position can be set to any number from 0 to 64 million.

b(nn) -- Goto Beginning: Go to the beginning of a subroutine nn or main program if nn not specified ("b" only). 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. Range of v is 0 to 9999 and must be placed between 0 and the last line of the subroutines.

l -- Learn and Store 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 effecter (by jogging, by using handwheel or manually) to that location, and issuing a "l" command. The net effect is that either to insert a new positional instruction or to replace current instruction with a new positional instruction. The generation of this positional instruction depends on two learning conditions, "learn" and "learn/replace". AT "lean" condition, it start with an empty program buffer and register a series of points; the other condition is to learn through a program already in the buffer and modify existing positions. SMCC detects its "buffer full bit", If buffer is not full, SMCC does a "learn", otherwise, it does "learn/replace" at whatever the point in the program buffer it may be.

[Example 4.2]

```
z           ;Clear buffer
F1024       ;Maximum Speed
HOME        ;Go to Home
DWE 1024    ;Stop for 1 second
j           ;At this point, move the end effecter
            ;to the desired position
j           ;Stop there
l           ;SMCC will learn the position
            ;and appropriate instruction will be
            ;written automatically.
DWE 1024    ;Wait for 1 sec.
J           ;Move to the next desired position
J           ;Stop Jogging
l           ;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 l command. Sometimes

you have to eliminate Y axis move command if you do not wish to move Y axis at that time.

z nn – Purge Buffer and reserve nn steps for program: If nn is not specified, purge program only. nn is 1 to 1790 (8K RAM) or 5630 (32K RAM). The rest of available memory can then be used to store subroutines (which is Labeled 00 to 63).

```
[Example 4.3]

z1790      ;Reserve all the available memory for main program
z          ;Purge program only.
```

s – Save: Save Buffer Program and i-parameters in permanent memory (EAROM). This command works only when the on-board DIP switch SW5 is set to True (ON), and SMCC is at the in-position state. The SMCC prompts error sign (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 i parameter is displayed. Corresponding buffer command format is Inn vv. See Appendix A. Glossaries of Initialization/Setup parameters.

Pnn = V – Set value for a variable parameter.: 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]

P01 = 10000     ;Variable P01 is defined.
X(P01)          ;This command is same as X10000
```

Cmm = AA -- Defines Cmm as address AA: This creates look-up table. The equivalent Buffer instruction is: SET Cmm = AA.

```
 [Example 4.5]

 C12=ffb8        ; Define C12 as Parallel Port Address (FFB8)
```

Mnn = Cmm.b – Defines M function bit at address Cmm at bit position b.
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 specifc M function bit can be set by SET nn and reset by RESET nn. See chapter 5.

Following commands are for Special Applications Only

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

```
[Example 4.6]

r2034,5          ;Will read Version Number
ra3a4,2          ;Will read End address of RAM Memory.
```

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

```
[Example 4.7]

da414,20         ;Will display User Program converted to
                 ;internal format and stored.
```

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, will send back one byte of data.

(2) All addresses are in hexadecimal format, using lower case a to f.

4.3. PROGRAM EDITING COMMANDS

LIST n,m -- List m lines of buffer program starting at line n.
LIST n -- List Line n only
LIST -- List whole program.

DELETE n,m -- Delete m lines of buffer program starting at line n
DELETE -- No Delete for Protection.

INSERT n -- Insert additional line(s) of program starting at line n. To exit, use (CR).

REPLACE n -- Overwrite program line n. To exit, 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 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 "znn" command is used, the entire program and the labels are cleared, and n lines (0 to 1790) are allocated to 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 main program area. n is independent of available memory. All labeled subroutines are moved automatically to create necessary program buffer space.

[Example 4.8] - Demonstrate editing commands

```
A0          ; select A0 card
z50         ; clear the program buffer and reserve 50 lines
            ; for main program.
FOR 3       ; start a loop, do 3 times.
T2048       ; specify a move time of 2 seconds
X1000       ; Tell X axis to move to 1000
DWE1024     ; wait one second.
            ; note: command can be abbreviated by its first
            ; three characters.
X0          ; return X axis to 0
DWE1024     ; wait one second
NEX         ; End of loop
END         ; End of program
REP0        ; SMCC does nothing, but places itself in edit
            ; mode.  REP is the editor's "REPLACE" command.
            ; Program line number only shows up in edit mode.
LIS         ; This is the "LIST" program command
            ; You will read 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 replace old line 1 and SMCC send new
            ; line 1 as follow::
            ; 0001:T1024
LIS         ; list 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.
LIS         ; list program again
            ; 0000:FOR3
            ; 0001:X1000
            ; 0002:DWE1024
            ; 0003:X0
            ; 0004:DWE1024
            ; 0005:NEX
            ; 0006:END
```

```
INS00       ; insert new line before line 0
T1024       ; this becomes line 0 and SMCC prompt it as
            ; 0000:T1024
LIS         ; list program again,
            ; 0000:T0124
            ; 0001:FOR0
            ; 0002:X1000
            ; 0003:DWE1024
            ; 0004:X0
            ; 0005:DWE1024
            ; 0006:NEX
            ; 0007:END
s           ; Ths "s" command will save the program
            ; permanently.
            ; Program remain unchanged until next "s"
            ; is issued.
```

NOTE: Commands such as "X0Y0", "U1000V-1000" (incolves X and Y axis) while been entered at one source line but internally it occupied 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.
Note: LCD display may not catch up the burst of text transmitted by ix command. In that case, lower the Baud rate or use individual inn commands.

f -- Send Following Error.
p -- Send Current Position.
i38 determines the way a value will be presented by p command. If i38 is in default value (i38 1:1.0), then SMCC displays position in terms of count.

v -- Send Current Velocity. Unit is determined by i17.
Velocity is displayed in terms of RPM unit, if i17 is set by
i17 = (30,000,000) / cm
where "cm" is count value corresponds to one revolution.

? -- Send Status and Error Codes
SMCC sends 8 bytes, each byte 8 bits, as status indication to the host. The status bytes are transmitted in Hex-ASCII format.

	BYTE 1 (FEB5)	BYTE 2 (FEB6)
Bit 0	Home Y Completed	Time Rate Mode
Bit 1	Home X Completed	Single Step
Bit 2	Continuous Move	Buffer Full
Bit 3	Block Request (MDI Mode)	Parabolic Mode
Bit 4	Auto Divide	Limit Switch
Bit 5	Feed Hold (HOLD)	Not Ready (X)
Bit 6	Running	Not Ready (Y)
Bit 7	Timer Active	Not Ready To Move

	BYTE 3 (FEB7)	BYTE 4 (FEB8)
Bit 0	Jog Positive (X)	Jog Positive (Y)
Bit 1	Jog Negative (X)	Jog Negative (Y)
Bit 2	Limit Position (X)	Limit Position (Y)
Bit 3	Following Error (X)	Following Error (Y)
Bit 4	I2T Limit (X)	I2T Limit (Y)
Bit 5	Not Stopped (X)	Not Stopped (Y)
Bit 6	Not In Position (X)	Not In Position (Y)
Bit 7	Zero Command Velocity (X)	Zero Command Velocity (Y)

	BYTE 5 (FEB9)	BYTE 6 (FEBA)
Bit 0	Edit Mode	Auto Divide Request
Bit 1	Insert Mode	Extended Move Flag
Bit 2	Extended Read	Extended Move Flag
Bit 3	Circle Mode	Home Done Flag
Bit 4	Wait X	CCW Circle
Bit 5	Wait Y	Circle > 180
Bit 6	Backlash X	Position Frozen (^V)
Bit 7	Backlash Y	Subr. Buffer Open

	BYTE 7 (FEBB)	BYTE 8 (FEBC)
Bit 0	Servo Pointer	Servo Pointer
Bit 1	Servo Pointer	Servo Pointer
Bit 2	Servo POinter	Servo Pointer
Bit 3	Dwell Request	Servo Pointer
Bit 4	Home in Progress	Servo Pointer
Bit 5	Dwell in Progress	Stop Request
Bit 6	Continuous Move Reqest	Servo Pointer
Bit 7	Memory Request	Servo Pointer

Bnn -- Send line number for Label nn, or size of program buffer if nn is not specified.
Bx -- Send line numbers at which all labels, 00 to 63, are located.

Pnn -- Send variable parameter values.
Px -- Send all variable parameter values for P1 to P15.

Cmm -- Send Defined Memory Location Address for Cmm.
Cx -- Send all defined memory location addresses for C00 to C63.

Mnn -- Send M function definitions.
Mx -- Send address and bit for all M functions, M00 to M63.

SMCC supports multiple number of M-function I/O's to interface with environments like limit switches, status of other equipments etc. Among them, several M-function I/O's are defined as Mnn variables.

```
Software    Description       Pin    Hardware
Definition                    Spec.
M01         ;Machine Output   #1     (MAO1)
M02         ;Machine Output   #2     (MAO2)
M03         ;Machine Output   #3     (MAO3)
M04         ;Machine Output   #4     (MAO4)
M05         ;Machine Output   #5     (MAO5)

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

```
GOTO+3 IF (M13=0)   ;If M13 (M-function Input #3) is FALSE,
                    ;skip next two instructions and go
                    ;ahead. Otherwise, execute next
                    ;instruction.
RESET 02            ;Turn Off M02 (M-function Output #2)
GOTO 05             ;Goto Subroutine 05
SET 01              :Turn ON M01 (M-function Output #1)
DWELL 1024          ;Wait 1 Second
  :
  :
```

[Example 4. 10] - Definition of Cnn and Mnn

```
                    ;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 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 receiption of this command, any on-line SMCC will immediately "freeze" 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 position data is captured at the time of <CNTRL V> command reception. It is possible to capture the position data simultaneously on all cards by 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

Following examples are intended to introduce basic SMCC commands that are most frequently used in many applications. It is recommended that motor shaft is disconnected to the load so that motor runs freely without hitting any limit. Be sure that the motor is tightly attached to a fixed structure, therefore, eliminate sudden jerk triggered by rapid change in acceleration.

Turn the power on, the SMCC holds motor shaft at a position and set that position to zero. Try to rotate the shaft (by manual), a well-tuned system should hold the position.

[Example 4.11] - Card Selection and Basic Move.

```
A0    ;Card Address Select Command.
      ;selects card #0. If only one SMCC card
      ;is used, that card must be card #0.
```

```
i38  ;Check if SMCC responds with 1:1.0
     ;if not, record the response for later use
i38 1:1.0 ;Example 4.17 has more detailed description on i38
J    ;Jog Forward. Motor will jog with
     ;predefined speed until commanded to stop.
     ;Memorize forward direction.
J    ;Typing J command again stops Jogging.
J    ;Again, Jog forward.
Q    ;Quit also stops motion.
=    ;It moves motor to prejog position.
j    ;Small j forces to jog backward.
     ;Next j will stop backward jogging.
p    ;This command asks current position. SMCC will
     ;respond with current position represented in
     ;counter unit. Relation to motor speed [RPM]
     ;or actual end effecter position will be described
     ;later.
v    ;This command asks current speed in counter
     ;unit per second.
q    ;It aborts current move. Note that SMCC controls
     ;or holds shaft position.
```

[Example 4.12] - Unit of Position.

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

```
k    ;This command kills servo action. Motor is not
     ;energized. You should turn the shaft freely.
     ;Set motor shaft in a known orientation so that
     ;you can refer it later.
     ;This command may issued whenever you notice
     ;unpredicted operation, and wants to turn off
     ;motor power.
p    ;Asks current position in counter unit. Take note
     ;of position value.
     ;Now, turn the shaft one revolution in a forward
     ;direction.
p    ;Again, asks current position. Subtract previous
     ;position value from current position value.
     ;Lets call this value to be "cm".
     ;This is the counter value corresponding to one
     ;revolution (mechanical).
q    ;This command forces motor to hold the position.
     ;released by k command.
p    ;Again, asks current position.
Z    ;It sets current position as Zero.
     ;Initially, the position is set to Zero,
```

```
        ;but can be set anywhere by this command.
p       ;This time, current position must show zero.
```

[Example 4.13] - Introduction to I-parameters, Positional Unit

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

One of the i command, i39 specifies Encoder Type, Direction and Position Counting Method. By typing "i39(CR)", get your current value 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,

cm = Encoder Lines per Rev. x 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 jog motion is specified by initialization parameters. Jog Speed is determined by two commands

i06 Set Reference Speed, and
i31 Set Jog Speed Ratio

```
i06     ;This command asks reference speed of SMCC in
        ;counter per second. Record down it for later
        ;use. You can convert this speed in RPM by
        ;the following equation.
        ;
```

```
                ;                        i06
                ; Reference RPM =     ------ x 60
                ;                         cm
                ;
                ;If i06 =20000, cm = 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 (i06).
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 :
                ;
                ;                                  i31
                ; Jog RPM =  Reference RPM x  ------
                ;                                  256
                ;
                ;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 shaft position as accurately as possible, SMCC compares desired position with current position obtained from the feedback sensor in about every milliseconds. the Following Error "FE" is defined by

FE = desired position - 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 nonzero Following Errors present, especially during the fast acceleration/deceleration. Also we must give some allowance to the system that Following Errors less than certain number of counts can be regarded as no error. This is mainly due to system's mechanical tolerance.

```
q               ;Stops motion.
f               ;Send Following Error in count unit.
                ;Since motion is stopped, Following Error
                ;must be close to zero.
j               ;Let Jog, and
f               ;during the acceleration or deceleration,
                ;you may notice some nonzero Following Error.
f               ;check FE several times.
q               ;Stops motion.
```

[Example 4.16] - Acceleration/Deceleration

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

```
i08         ;Asks specified acceleration time
i08 4096    ;Accel/Decel time in 4 second.
j           ;Let's see the slow change of speed.
j           ;Stop
i08 256     ;Accel/Decel time in 0.25 second.
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 position or distance was referred with counter unit. 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

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

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

```
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 end effecter is connected by a 5 pitch screw. One inch movement of the end effecter 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, SMCC will respond with 1.000 upon p command.

```
i38 10000:1000.3 ;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 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.

```
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 asks 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          ;Stop 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, 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 * in front of each instruction.

[Example 4.19] - Usage of Buffer Instructions

```
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 resets
                 ;to 0, after "Q", or "S" command is given.
                 ;As well as "i02 0" command.
*     F 1024     ;Feedrate(Speed) Instruction.
                 ;                          nnnn
                 ;      Speed = i06  x  --------  (Count/Sec)
                 ;                          1024
                 ;
                 ;Programmed Move after this instruction will be
                 ;executed with this speed.
                 ;Speed 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

SMCC uses two different kind of memory for storage. Initialization parameters 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 non-volatile memory called EAROM (or EEPROM). Data in EEPROM can be changed by s (save) command, but data are preserved during power down. Upon initialization, these

data are recovered to RAM to be used.

```
ix          ;This commands 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 SMCC

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

1. Hardware RESET - Press INIT Button on the front panel will reset SMCC completely (same as Power-up RESET). If multiple SMCC's are connected in parallel, all cards will be reset simultaneously.

2. $ Command (Software Reset) - This command re-sets 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 SWITCH Setting (Baud Rate, Address etc.). All Memory contents will be lost.

[Note] Commands or Instructions longer than 3 characters without counting modifiers or parameters, can, in general, be abbreviated to first three characters. For example,

```
DEL3        for DWELL 3
REP4        for REPLACE 4
DWE1024     for DWELL 1024
STA         for START.
```

[Example 4.21] - Control running a motion program

```
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).

5. BUFFER INSTRUCTIONS

Buffer instructions are stored in the buffer and then be executed later upon the reception of commands such as "S" (step) or "R" (run), or 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. Continuous Motion Block is defined by starting with START command, and ended with STOP command. SMCC will execute motion blocks consecutively without stopping. The effect of "START" can be ended by using "STOP" or "END" (if at end of program) or issuing an "S" step command.

STOP -- End of Continuous Motion. See description for START instruction.

END -- End of program. This instruction must be given at the end of Program. Otherwise "s" (save) command may not work. This also closes all the unclosed loops, or blocks.

Lnn -- Label defines subroutine from L00 to L63. A subroutine may contain any number of program steps and may contain loops and other subroutines, can be nested up to 16 levels deep. Each subroutine must be terminated by a RETURN instruction.

RETURN -- Return from Subroutine. Subroutine is defined by a Lnn command. If the Subroutine is executed through GOSUB command, program sequence will return to the next command in the main program, but if it is executed by GOTO command, 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 real application, The whole process involves only reassigning a value to a variable.

```
[Example 5.1]

SET P01=512          ;Define P01
FOR 10
F(P01)               ;Use defined variable
X10000
X0
NEXT
```

The parameters, P01 to P15, may be used in instructions that requires simple data such as X (Pnn) (move X axis) or Y (Pnn) (move Y axis) or F(Pnn) (Get Feed Rate), etc. However, Pnn parameters may not be used with GOSUB, GOTO and instructions that has no data (such as STOP, END, .. etc). Variables arithmetic operation are also possible within the instructions. Acceptable forms are:

```
1.  Instruction (+/-Pnn)
2.        "       (Pnn +/- Pmm)
3.        "       (+/- Pnn * Pmm)
4.        "       (+/- Pnn/Pmm)
5.        "       (Pnn + Pmm * Pkk)
6.        "       (Pnn +/- Pmm/Pkk)
7.        "       (Pnn + Pmm*Pkk/Pjj)
8.        "       (Pnn * Pmm / Pkk)

valid usage such as
SET P8 =(P06*P11)
SET P7 = P8 + 1
SET P1 = 5
SET P2 = (P1)
SET P1 = (P7*P10/P11)
```

The following commands can incorporate with the P parameters i.e, their data can be replaced by the P parameter.

```
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 P parameters may be effectively used in applications such as:
1) Peck Drilling.
2) Generating spirals.
3) Generating 2nd, 3rd and higher order polynomials for arbitrary path generation.

SET Pn = HX
SET Pn = HY
SET Pn = AX
SET Pn = AY These instructions set the variable Pn 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 encoder counts.

RESETP -- Causes all P parameter values to be set to zero.

FOR nn -- Start loop and execute the loop nn times. End of loop must be specified by NEXT command. If nn = 0, then loop is executed forever. Range of nn is from 0 to 32767. Loops may contain other loops (nested FOR), as long as each loop is terminated properly by a NEXT command. Loops may also contain instructions that

execute subroutines.

[Example 5.2]

```
FOR 10          ;Outer loop  will be executed 10 times.
DWELL 1024      ;Pause 1 Second
                ;
FOR 2           ;Forward and Backward Motion for 100
                ;counts will be executed 2 times.
U100
U-100
NEXT            ;Corresponds to FOR 2
NEXT            ;Corresponds to FOR 10
```

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

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

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

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

GOSUB nn (IF (NOT) Condition) -- Call subroutine nn (defined by Label Lnn) and return to next program step when RETURN is executed.

Notes: "Condition" format in the above instructions must follow one of two forms.

(1) IF(NOT) nn CC xxx -- Memory Contents with Immediate Data.
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 following forms:

"=" (EQUAL)

">" (GREATER THAN)
"*" (BITWISE LOGICAL AND)
"+" (BITWISE LOGICAL OR),

[Example 5.3]

```
C01 = ff3a              ;Define C01 as Address FF3AH(Port 0
                        ;Buffer)
GOTO 05 IF 01 * 16      ;If X axis Positive Limit is Set,
                        ;then jump
HOME                    ;to Subroutine 05. Otherwise,
                        ;Execute HOME.
```

(2) IF(NOT) cc (AND dd)
IF(NOT) cc (OR dd)

cc or dd denotes a conditional test that can have one of two forms:
a. (Mnn = 0/1), where Mnn is a bit defined by Mnn = Cnn.b.
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]

```
C04 = ffb7      ;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 MI1 is HIGH and MI2 is low then execute
                ;Subroutine 02 defined by L02.
GOTO+ 05 IF NOT (P11 > X)
                ;If present position (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 description of GOSUB Command for the format of Conditions.

SET Cnn = AA -- Defines Cnn as address location AA. Range of nn is 0-63 and creates an address look-up table. Once address is defined, it is referred by the Mmm instruction or the CASE instruction in the subsequent program.

SET Mnn = Cmm.b -- Defines address location Cmm, bit b as input or output bit defined by Mnn. Mnn is (0-63). Cmm is (0-31) and b is (0-7).
See Appendix D for default definitions.
[Example 5.5]

```
SET 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, 0-63).
xxx Defines the size of the look-up 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 have 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 entry (instruction) can be as many as 255 steps and are written immediately after the CASE. If the data value read is larger than the table, the table is bypassed and the next instruction is executed.

The instructions in the table can be any instructions, including: Moves, Subroutine GOTO's 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. Variable Cnn (nn = 0 - 63) is specially designated to define memory address.

```
C20 = ffb8      ;Define C20 as parallel port.
CASE 20,10L     ;Read Low Nibble (L: Lower 4 bit) of C20
                ;and jump to the following 10 tables
                ;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 subroutine, program
          ;jumps to here.
SET M01   ;Turn ON M-function Output #1
```

Following instructions are for Special Application 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 changing parameters during program execution.

X = nn [Y = nn] -- Set Current Position as given by nn. The unscaled range (i38 set to 1:1.0) is –67108864 to 67108863

Fnn -- Set Feed Rate.
Range is from 0 to 8 million (2*23-1).
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 is a modal and defines the move time of the X and Y axis (from start to beginning of deceleration point) 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.

tnn -- Set Speed Change Time in nn/1024 Seconds. Equivalent to i08 (Set Accel/Decel Time) Initialization Commands. It supersedes i08 for programmed moves, but does not change i08, and does not affect jog moves. The total time of a parabolic move segment is 't'

TORQUE -- This puts SMCC in torque-limited mode.
If it is followed by a "HOME" command, then the home cycle changes to a torque-limited home. The home move continues until the Following Error on each axis exceeds 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 up 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, then the command position for that move will not be relaxed if the Following Error is greater than i10.

LINEAR -- Defines Linear Motion (Modal Command) All subsequent instructions are executed in linear mode until a CIRCLE instruction is executed.

CIRCLE n -- Defines CW/CCW Circular Motion (Modal Command).

```
n = 0 :  CW Direction with angle less than 180 degree.
n = 1 :  CCW Direction with angle less than 180 degree.
n = 2 :  CW Direction with angle greater than 180 deg.
n = 3 :  CCW Direction with angle greater than 180 deg.
n = 4 :  3-point smoother (S-Curve)
```

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 in Parabolic only. Both X and Y radius must be defined before executing any move instructions (can be defined before or after the CIRCLE instruction).

R(X) = nn Set the present X radius to nn.
The range of nn depends on the scale in effect.
The unscaled range is 0 to nn (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 circular arc from present position.

```
F1024           ;Define Feedrate
CIRCLE 0        ;Define whether the motion is CCW or CW.
R=1000          ;Define Radius
X1000Y1000      ;Define Final Arc Point.
```

Note:
As can be seen from the above example, Arc is completely defined by Specifying 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 circle starts at 0/0 goes to 10,000/0 (180 degrees) and then returns to 0/0 to complete the circle.

[Example 5.8] - Circular Interpolation between X axis on A0 and Y axis on A1

```
AAz
F1024
R=10000
CIRCLE 0
A0 U20000WY0    A1 WX20000V0
A0 U-20000WY0   A1 WX-20000V0
END
```

For reference, to do interpolation on a single card, between X and Y axis, the program would look as follows:
To move elliptic arc from present position, program as

```
A0z         ;Address Card A0
F1024       ;Feedrate to 100%
R = 10000   ;Radius of 10000 encoder counts
CIRCLE 0    ;Clockwise Circle.
U20000V0    ;Incremental move of X axis to 20000 and Y
            ;axis of 0. Half Circle to 180 degrees.
U-20000V0   ;Return to 0
END
```

The differences 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 X axis on A0 and X axis on A1

```
AAz
```

```
F1024
R = 10000
A0 CIRO A1 CIR1      ;Note that A1's X axis must be reversed,
                     ; using CIR1.

A0 U20000WY0   A1 U0WY20000
A1 U-20000WY0  A1 U0WY-20000
END
```

[Example 5.10] - To move elliptic arc using A0, X and Y axes from present position, program as

```
F1024           ;Define Feedrate
CIRCLE 0        ;Define whether the motion is CCW or CW.
RX=1000         ;Define X Radius
RY=2000         ;Define Y Radius
X1000Y2000      ;Define 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.

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 default is the value of i07, or 65498. And mm is a slope increment number added to feed rate each servo cycle. Its range is (0 - 65535), and 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 Home position. Upon HOME command, shaft moves to the direction defined by i24(Set Home Direction), with the speed defined by i04 (Home Feedrate)

until corresponding switch conditions (set by i26) are met. SMCC then moves to offset position defined by i25 (Set Home Offset) from that position. 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 home position by using the "HOME1" (X axis) or "HOME2" (Y axis) Commands.

DWELL nn – Stop and hold position for nn/1024 seconds.
Range is 0 to 8 million (2**23-1).

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

AX nn [AY nn] -- Move X[Y] axis to the position nn units from Home or initial zero position. (Absolute Move relative to Home.)
The unscaled range is - 67108864 to 67108863.

U nn [V nn] -- Move X[Y] axis nn encoder counts from the current position (Relative Move).
The unscaled range is - 67108864 to 67108863.

WX nn [WY nn] -- Wait time (no move) equal to equivalent time to 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 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,V) to define parabolic move with ending velocity 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 lines 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 received a "S" (step) or "R" (run) command before a motion command is sent to the amplifier. 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]

```
;"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.
:
.
```

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:

```
:              ;"S" (step) command is issued
.
X5000          ;Move completes then "S" issued again
DWE 20         ;Dwell is completed, "S" issued again
               ;Use a dwell of at least 20 to provide
               ;enough calculation time.
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 trigger is received.
:
.
```

This sequence would work correctly, and not cause any unwanted jerks since the DWE 20 (DWELL) forces the X5000 move to be completed and then allows the "WUT" to be executed. Note that if instead of "S" (step) commands, a "R" (run) command was used at the start of the sequence, then X5000, DWELL 20, and the WUT commands would execute automatically and the X10000 move would be precalculated and 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" as the case may be, 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 passing 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 precalculate 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 i26 to configure trigger 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 trigger 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 wil 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 a value is used for "tt"

q... (Parabolic version 2.79 and newer)
Altered Destination Move In MDI Mode

Sometimes when commanding 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 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]

```
i021            ;MDI mode - execute moves immediately
F100
X10000          ;start move toward X = 10000
qX-5000         ;If issued while previous move is still
                ;executing
                ;this will cause SMCC to break into that
                ;move and alter the trajectory to proceed
                ;to this new 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 Program

[Example 5.14] Usage of Buffer Instructions

```
z          ;This command empties buffer for main program
           ;znn command will purge buffer and reserve space
           ;for nn program steps.
Z          ;Reset Current position to Zero.
i381:1.0   ;Default Positional Scale
i02 1      ;Specifying i02 = 1 makes Buffered instructions
           ;processed immediately.
           ;Note that this parameter is automatically resets
           ;to 0, after "Q", or "S" command is given.
           ;As well as "i02 0" command.
F1024      ;Feedrate(Speed) Instruction.  Actual Speed is
           ;for Fnnnn,
           ;                      nnnn
           ;Speed = i06    x  ----------     (Count/Sec)
           ;                      1024
           ;
           ;Programmed Move after this instruction will be
           ;executed with this speed.
           ;Speed 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 specified 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 specify the 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 5.15] Time Specified Move, and Fictitious move

Assuming cm = 2000
i06 60000 (1800 RPM)
i08 1024 (1 sec.)

Following two programs will give almost identical motion trajectory

```
                ;*** Time Specified Move ***
Z               ;start from Position Zero
T 5120          ;Move will be finished in 5120/1024 Seconds.
                ;Note that specified time is from start to
                ;end of constant velocity move. So actual
time
                ;will be 5120/1024 + Deceleration time.
X100000         ;100000 count distance will be moved
                ;in 6 seconds (5 + 1 Decel time)
                ;Speed will be 100000/5 = 20,000 Count/Sec.
DWELL 1024      ;Pause 1 sec
X150000         ;This time, 50000 count distance is
                ;moved in 6 seconds. So speed is in half 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          ;t instruction specifies accel/decel time
                ;like i08 command.
                ;time unit is in 1/1024 sec. (roughly
                ;in millisec.)
U 100000        ;remember, this is a relative move
                ;Move 100000 counts backward.
                ;This time, accel/decel in 0.5 seconds.
                ;with full Reference speed.
DWE1024
F170            ;Reduce Speed in Half
```

```
U 50000
WX160000        ;Wait amount of time corresponding to execute
                ;X160000 (10000 Move)
END             ;End of Program
```

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

[Example 5.16] More Instructions, Loop Definition

A Loop can be defined by FOR -- NEXT instructions. A program block inside loop will be executed as many times as defined in FOR instruction.

```
HOME            ;Go to Home Position
                ;Note that small "h" is the corresponding
                ;Direct Command.
F 1024          ;Set Speed.
FOR 3           ;Start of a Loop. Do this block 3 times.
                ;(FOR 0 lets program in infinite loop.)
DWELL 1024      ;Wait 1 Second (Time unit is same as in T
                ;command)
AX1000          ;Absolute move from Home Position.
AX0             ;Go back home.
NEXT            ;End of FOR loop
END
```

[Example 5.17] Usage of Subroutine

Subroutine is defined by starting with a Label command (L nn) and ended with 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 command.

```
                ;This is main program

F 512           ;Speed Definition
GOTO + 01       ;Relative Jump
U10000          ;Because of GOTO + 1 Command,
                ;this instruction skipped.
U20000          ;Move 20000 counts
GOSUB  00       ;Go to Subroutine 00 and come back.
DWELL  1024     ;Wait 1 second.
GOTO   00       ;Go to Subroutine and END at RETURN command.
                ;Do not come back.
END             ;End of Main Program
                ;
                ;Subroutine Starts Here.
L00F1024        ;Definition of Subroutine 00.
```

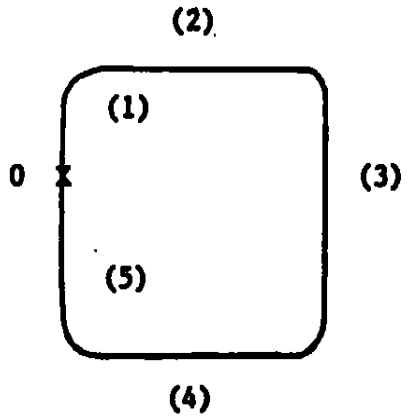
```
                ;and Speed Declaration
X0              ;Go back to Position 0.
GOSUB 63        ;Subroutine Call inside a subroutine
RETURN          ;End of Subroutine 00.
L63RET          ;Dummy Subroutine
```

[Example 5.18] Dual Axis Coordinated Motion

SMCC can control two axes if 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 execution of the next block is started when current motion starts decelerating, following example draws a square with round corner.

[Example 5.18a]

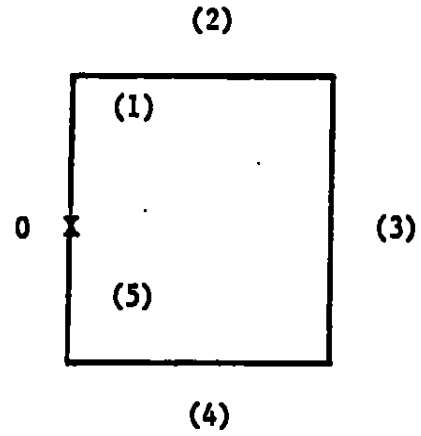
```
Z           ;Not a Buffer Instruction. Sets current Position
            ;Zero.
F 1024      ;Feed rate Maximum.

Y 25000     ;Move Y axis     (Path 1)

X 50000     ;Move X axis     (Path 2)

Y -25000    ;Back Y axis     (Path 3)

X 0         ;Return X axis   (Path 4)

Y 0         ;Return Y axis   (Path 5)

END         ;End of Program
```



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

[Example 5.18b]

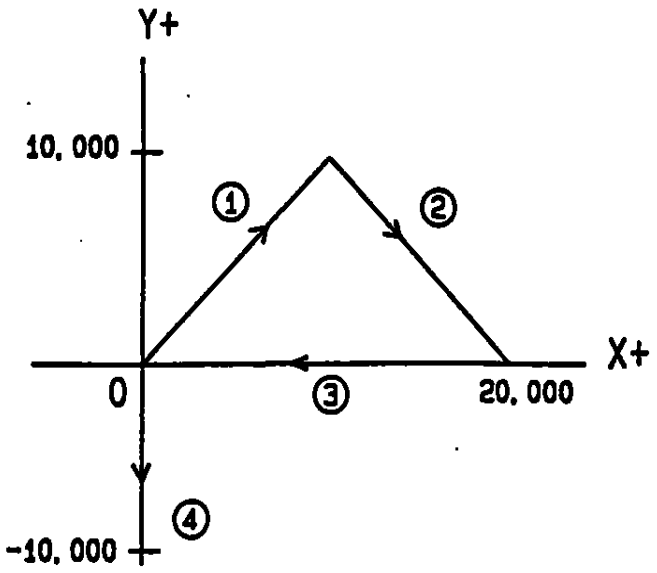
```
F 1024      ;Feed rate Maximum.

Y 25000     ;Move Y axis     (Path 1)
DWE 0
X 50000     ;Move X axis     (Path 2)
DWE 0
Y -25000    ;Back Y axis     (Path 3)
DWE 0
X 0         ;Return X axis   (Path 4)
DWE 0
Y 0         ;Return Y axis   (Path 5)

END         ;End of Program
```



[Example 5.19] Double Axes Instructions

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

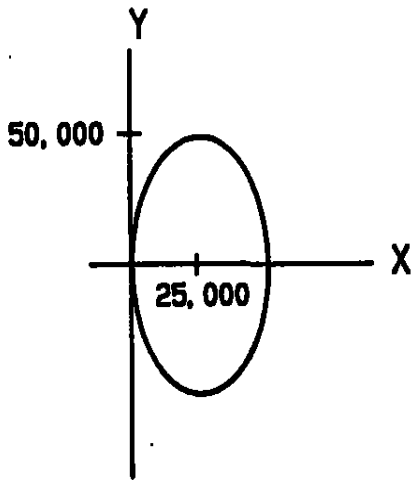
```
F1024                ;Full Speed
X 10000 Y 10000      ;X Y axes move simultaneously.   (1)
                     ;45 degree diagonal Motion.
U 10000 V-10000      ;Relative simultaneous Motion    (2)
DWELL 1024           ;After motion finished,
X 0                  ;Move X axis Back                (3)
DWELL 0              ;
Y 10000              ;Move Y axis Back                (4)
END
```



[Example 5.20] Elliptic Motion.

Elliptic motion can be defined with two principal radii. Following example draws an ellipse.

```
F 100          ;Define 1/10 Speed
CIR 1          ;Define Counter Clockwise Direction
RX = 25000     ;Define X radius
RY = 50000     ;Define Y radius
U 50000        ;Move to 180 degree point.
U -50000       ;Move back
END
```



[Example 5.21] Multiple SMCC Coordinated Motion

Suppose A0 and A1 cards are connected in parallel to the host computer. Knowing that Card address command can be placed anywhere in the instruction, to draw a diagonal line,

```
AA                          ;All card addressed
Z                           ;Clears all buffers
T 2048                      ;Timed Move in 2 Seconds.
A0 X 10000 A1 X 5000        ;X Y axes move simultaneously.
                            ;26.5 degree diagonal Motion.
A0 U 10000 A1 U-5000 AA     ;Relative simultaneous Motion
END

A0 LIST                     ;Verify A0 Program
```

```
A1 LIST                    ;Verify A1 Program
AA R                       ;RUN
```

[Example 5.22] Variables defined Directly.

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

```
T10240          ;Move in 10 second.
U(P05)          ;to the amount defined by P05.
END
```

To run the program, you can issue commands as

```
P05             ;Direct command to ask stored definition.
P05=500000      ;Direct Command to define Variables.
Px              ;Will List whole 15 variables with stored
                ;parameters.
R               ;Run command.
```

Many Instructions allow use of Variables instead of Constants. Also some arithmetic capabilities in 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

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

```
M01 ;Machine Output #1 (MAO1)
M02 ;Machine Output #2 (MAO2)
M03 ;Machine Output #3 (MAO3)
M04 ;Machine Output #4 (MAO4)
M05 ;Machine Output #5 (MAO5)
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)
```

Following instructions are one of the possible usage for Mnn variables.

```
GOTO+03 IF M13 = 0   ;If M13 is FALSE, skip
                     ;next three instructions and go ahead.
                     ;Otherwise, execute next instruction.
RESET 02             ;Turn Off M02 (M-function Output #2)
DWELL 1024           ;
GOTO 05              ;Goto Subroutine 05
SET 01               ;Turn ON M01 (M-function Output #1)
DWELL 1024           ;Wait 1 Second
   :
   :
L05
   :
   :
```

[Example 5.24] Processor Address Definition (Bits and Bytes)

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

```
                ;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.
                ;Definition inside Program
SET C01 = ff01 ;Define C01 variable as SMCC memory
                ;address FF01.
SET M01=C01.5   ;M01 is defined as bit 5 of that address.
```

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

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. PLC program can not directly perform any motion, but it can change variable parameters by logical comparison of Position, Velocity, following error, and variable parameters (Mnn, Cnn, Pnn). SMCC can add more powers in many applications with 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 normal buffer program. PLC Programs are active at all times, even if a motion program is not being executed.

To enable PLC Program set i62 to 1 (Parabolic), default is PLC disabled.

6.1 Direct PLC Commands

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

PURGE -- Purge PLC Program
This command deletes PLC program stored in memory. PLC program cannot be edited. To modify PLC program, "PURGE" it, and reload modified program. To expedite program editing, Invokes SMCC MONITOR Program and use its built-in full-screen editor

6.2 PLC Buffer Instructions

IF Condition -- Start of Condition, see Note 1

THEN Action -- Do this if condition true, see Note 2

ELSE Action -- Do this if condition not true

AND Condition/Action -- AND previous condition or Action

OR Condition/Action -- OR previous condition or Action

END -- End of PLC program

A nested IF-THEN-ELSE can construct even more complicated PLC program. In general, PLC branches can also be constructed by using AND IF, THEN IF, ELSE IF.
The IF statement places SMCC automatically in PLC programming mode.
A bad PLC program will trap SMCC, it is best to save PLC programs in EAROM only after they have been fully debugged.

Every IF condition requires a THEN statement 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 SMCC will add them automatically, However, if one of the ELSE statement 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-type statement carried a condition with it, not an action.

Following are valid PLC instructions.

```
IF CO1 = 10
THEN P03 = 1024
END

IF FEX > 2000
OR VX < 20000
OR P05 = 2048
THEN P05 = P05 + 2048
AND M06 = 1
ELSE P05 = 1024
AND I08 = 256
END

IF M01 = 0              ;If M01 and M02 are true, then
AND M02 = 0             ;
THEN I02 = 1            ;Sets 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
ELSE.....
```

Notes (1) Condition Formats:

(a) Mnn = 0/1 depends of the status of M-function I/O. nn = 0 to 63.

(b) Cnn = vv depends of the byte value vv of SMCC address defined by Cnn = (or SET Cnn=) command. nn = 0 to 63, vv = 0 to 255.

(c) Tn = 0 True if timer, n = 0-3, has counted down to 0.

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

```
Cnn,m      nn = 0-63, m =1-4, m bytes of data defined by Cnn.
AX/AY      X or Y absolute position
VX/VY      X or Y velocity value
FEX/FEY    X or Y following error
Pnn        Pnn parameter value
```

In format (d), the "=" (equal to) cannot be used immediately follow the "IF" statement. In order to check for equality, use the "AND" evaluation.
for example:

```
IF P6>4    ;these statements check the equality P6=5
AND P6<6   ;which should not be coded in PLC programs
THEN ..    ;directly
```

Note (2) Action Formats:

(a) Mnn = 0/1
Set or reset M-function output.

(b) Cnn,m = Cnn,m +/- VVVV
Add a constant VVVV.

(c) Inn = VVVV
Modify i parameter.

(d) Pnn = Cpp,q
Assign or modify a variable with address data.

(e) Pnn = VVVV
Assign or modify with constant.

(f) Pnn = Pnn +/- VVVV
Increment or decrement Pnn by VVVV.

(g) Cnn,m = VVVV
Set memory location Cnn, m bytes to a value VVVV. 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 InnVV ... etc.

(j) Tn=SSSS
Assign timer n, n = 0-3, data SSSS has range of 0-65535.

(k) Pn =ATAN (value)
Assign the arctangent of value to parameter Pn, 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) continuously.

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 variable parameters (+/-P1), 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 x 43.69 = 21.845 seconds and then test for Time Out to occur by using:

```
IF T1 = 0
THEN.......
```

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 below:

```
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 SETTING AGAIN.
IF   T0=0
THEN M01=1       ;TURN ON M01 WHEN TO HAS TIMED OUT.
IF   T1=0
THEN M02=1
IF   T2=0
```

```
THEN M03=1
IF   T3=0
THEN M04=1
ELSE
END
```

The timer 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 Main Program

```
;SMCC Buffer Program

FOR 0                  ;Do Forever
F 1024                 ;Full Speed
FOR 100                ;Do 100 times
GOTO 03 IF(P05 = 1)    ;If P05 = 1, Jump to Subroutine
CIR 0                  ;CW Circle
R = 10000              ;Of radius 10000
X 20000                ;First half-circle
X 0                    ;Next half-cycle
END

L03                    ;Subroutine
HOME
RETURN

;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
```

[Example 6.2] COUNTER USING M-FUNCTION INPUT (COUNTER):

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

```
A0
PURGE                   ;Clear PLC program
IF M11=0                ;Check if M11 switch is toggled
                        ;M30 is used as flag to cause
                        ;the P1 counter to only count
                        ;once when M11 switch toggles.
THEN IF M30=0           ;Check if M30 flag is set
THEN M30=1              ;True, reset M30 flag
AND P1=P1+1             ;Add 1 to P1 counter
AND "P1"                ;Display value P1 on the screen
ELSE                    ;
IF M11=1                ;Check if M11 switch is released
THEN M30=0              ;True, set M30 flag
AND IF P1>999           ;Check if P1 counter has
AND P1<1001             ;reached 1000 counts.
THEN P1=0               ;True, reset counter
AND SEND"1000-Counts"     ;Send a message to the screen
ELSE
END                     ;End of PLC program
```

[Example 6.3] Use M-FUNCTION INPUTS AS CONTROL PANEL INPUTS:

This PLC program uses 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.

```
A0
PURGE                   ;Clear all existing PLC programs
IF M15=1                ;Check if M15 switch is toggled
THEN IF P1<1            ;Check if send flag 1 is set
AND P1>-1
THEN"R"                 ;True, starts continuous program
                        ;execution
AND P1=1                ;Reset send flag 1
ELSE                    ;
ELSE P1=0               ;Set send flag 1
IF M16=1                ;Check if M16 switch is toggled
THEN IF P2<1            ;Check if send flag 2 is set
AND P2>-1
THEN"Z"                 ;True, set X and Y position to 0
AND P2=1                ;Reset send flag 2
ELSE                    ;
ELSE P2=0               ;Set send flag 2
IF M17=1                ;Check if M17 switch is toggled
THEN IF P3<1            ;Check if send flag 3 is set
AND P3>-1
THEN P3=1               ;True, reset send flag 3
AND"hX"                 ;Starts  X axis on home cycle
ELSE                    ;
```

```
ELSE P3=0              ;Set flag 3
END                    ;End of PLC program
```

[Example 6.4]

This example shows the usage of a PLC program and suggests a way to organize the writing of PLC program. This example demonstrates the use of PLC for providing a function which allows a single switch to be used as 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 SMCC.
This program uses of two servo status bits to determine whether to use the single switch as a homing flag or a Travel Limit switch. This will be explained in the program comments.

```
A0z                ;Address Card & Clear Program Buffer
SET C07=feb6       ;Define C07 to be address feba (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 feba (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).
HOM                ;Start a home cycle. Choose and set appropriate
                   ;parameters for homing, i04 for speed, i24,25
                   ;and 26 X axis, i44, 45 and 46 Y axis. Homing
                   ;should be in a direction that will drive it
                   ;toward 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 a Loop forever.
F1024              ;Set 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 motion program. Note that a program "end"
                   ;is used for the program itself.
                   ;At very end, another "END" is used for the
                   ;PLC program.
A0
PUR                ;Address Card and Purge all the PLC programs
                   ;stored in the PLC 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 PLC is always active
                ;unless turned off, and that i62 simply get
                ;executed on-line and are 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 condition M51=0
                ;meaning: Look to see if there is an interrupt
                ;operation in progress, (The Home Command sets
                ;ffca bit 6 low.) If there is, then set feb6
                ;bit 4 high (M50=1) thus disabling the limit
                ;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, SMCC
                ;will automatically insert it.
END             ;The END statement closes PLC programming. If
                ;there are other PLC statements simply continue
                ;with an "IF" which opens a new PLC statement or
                ;branch until all PLC statements are completed.
```

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

	PARABOLIC SMCC	
	X axis	Y axis
Limit Status	feb6.4	feb6.4
Interrupt Status	ffca.6	ffc8.6

Finally, a counter PLC program is constructed as follows:

```
                ;Counter, using M Function
                ;input. (counter)
                ;The Counter is triggers
                ;on the trailing edge of the
                ;MI1 input
AOPURGE         ;Clear PLC program
IF M1=0         ;Check if MI1 is toggled
                ;M30 is used as flag to
                ;cause the P1 counter to
                ;only count once when MI1
                ;toggles.
THEN IF M30=0   ;Check if M30 Flag is set
THEN M30=1      ;True, reset M30 Flag
```

```
AND P1=P1+1      ;Add 1 to P1 counter
AND "P1"         ;Display value of P1 on
                 ;the screen
ELSE             ;
IF M11=1         ;Check if MI1 sw. is released
THEN M30=0       ;True, set M30 Flag
AND IF P1>999    ;Check if P1 counter has reached
AND P1<1001      ;1000 counts
THEN P1=0        ;True, reset counter
AND SEND "1000-counts"   ;Send a message to the screen
ELSE
END              ;End of PLC program
                 ;Note that the counter may be
                 ;preset by setting P1 to a
                 ;starting value.
```

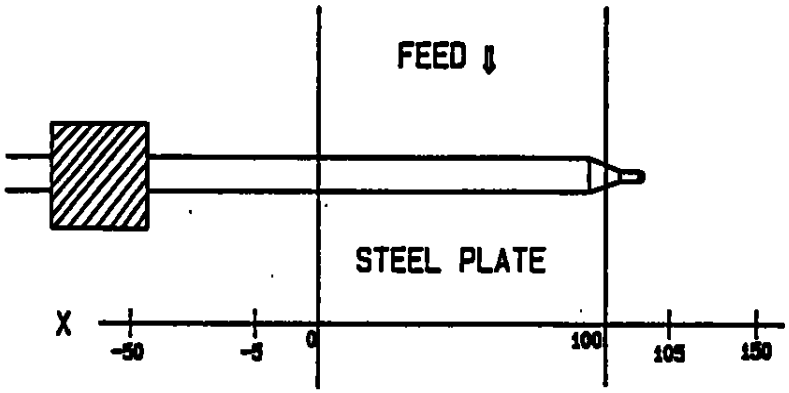
7. APPLICATION PROGRAMS

[Example 7.1] Single Axis Torch Cutter

Suppose a SMCC system is installed to operate a Torch Cutter. Driving system includes a Baldor motor to a gear train of 4:1, and a 5 pitch Ball Screw to convert to linear motion. Motor has 500 Line encoder as feedback devices.

The application system has following requirements:

a Maximum Travel Speed (No Cut): 180 IPM (Inches per Minute)
b Cutting Speed: 45 IPM
c Maximum Acceleration: 180 IPM/Sec. = 3 Inches/Sec.Sq.
d Cutting width: 100 inches
e Allowed Range of Working Position +150 inches, -50 inches from Position Zero
f Handshake:
M11 (MAI1): Feeder Ready Input (1=Ready, 0=Not Ready)
M01 (MAO1): Torch ON/OFF Command (1=OFF)
M02 (MAO2): Raise/Lower Tip Command (1=UP)



1. Pre-calculation

a One Rev. = 2000 counts
b One inch Move = 20 Rev. = 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 inches move)
f Travel during Accel to Cut Speed = 45/60 inch move.

2. Alignment and Setup

Position zero is set at the position indicated in the above diagram. Range of cutter operation will be -5 to 105, to allow enough travel to accel/decel.

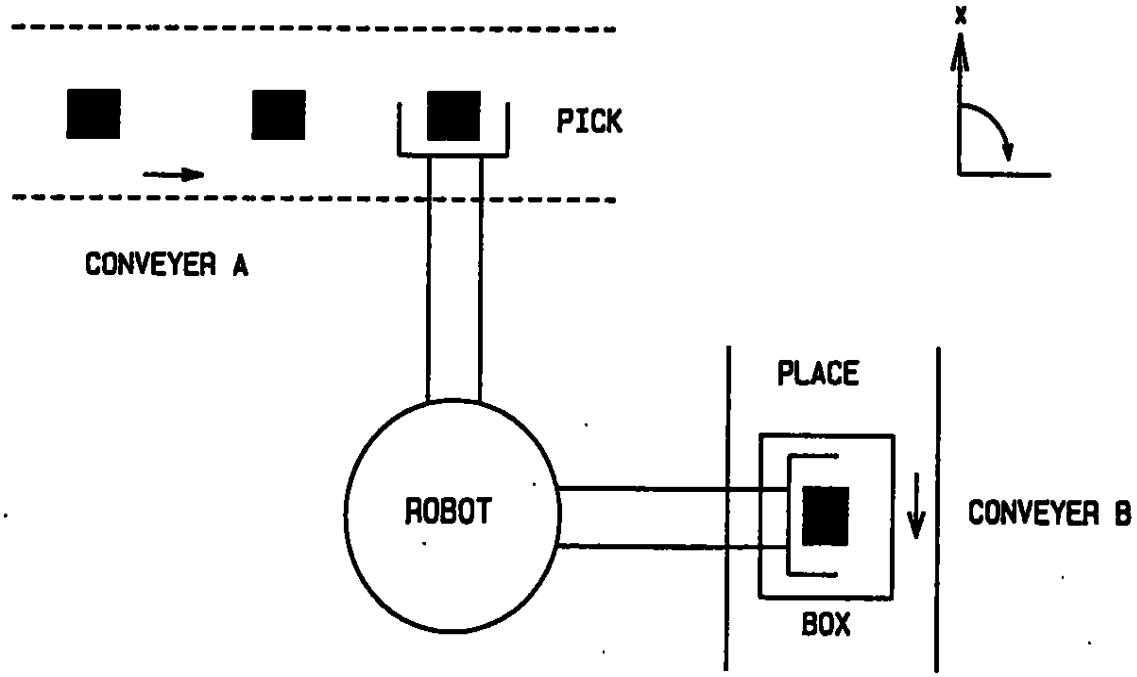
```
i38:40000:10000.4     ;Position in Inches
Set Position Zero
```

3. Program

```
F1024                 ;Start Position
X-5.0
RESET 01              ;Torch ON
FOR 20                ;Number of Sheets to be cut
SET 02                ;Tip UP
F1024
X105.0                ;Move to Start Position
DWELL 100
GOTO-01 IF (M11=0)    ;Wait Until Feeder Ready
RESET 02              ;Tip Down
F256                  ;Cutting Speed
X-5.0                 ;Cut
NEXT
SET 01                ;Torch OFF
END
```

[Example 7.2] Two DC Motors in Pick and Place Operation

Suppose a SMCC system is installed to control "Pick and Place" Robot Arm. The Robot is configured as Two-dimensional Cylindrical Coordinate, and has a gripper to grasp and release parts. Refer to following diagram.



Its task is to grasp a part from conveyer A, carry it to the top of box on another conveyer B, and drop it. 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 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
```

Programming requirements are as follows.

a X axis unit in inches, Y axis in degree.
b Max. Speed motion when no-load, 1/4 speed with load.
c 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 Initial Setup and Tuning is properly done with the following parameters.

```
i38 50000:10000:4   (X Unit in inch)
i58 72000:10000:4   (Y Unit in degree)
P01 = 100   Job Order issued
Ref. Speed and Acceleration is set appropriately.
```

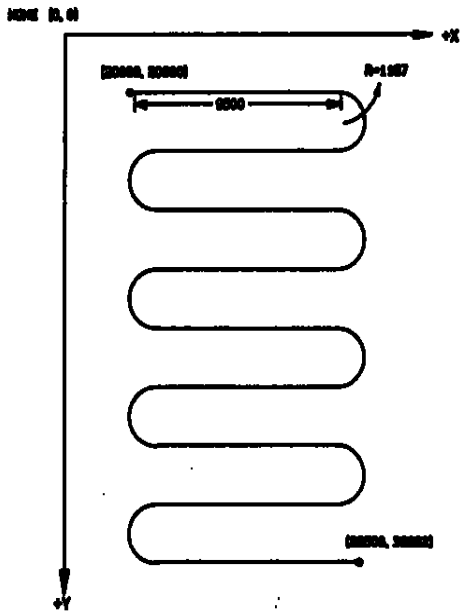
2. Program - After setting Absolute Position, and Job Count command, Operator can RUN ("R") the following program.

```
SET P03 = 0          ;Initialize Job Count
SET 01               ;Gripper Release
RESET 02             ;Turn OFF Bell
F1000
X25.0Y0.0            ;Move to Rest Position
FOR (P01)            ;Specified Number of Times
SET P02 = 0          ;Reset Time Count
F1000                ;Fast Speed
X25.0Y0.0            ;Go to Rest Position
DWELL 100
GOTO-01 IF(M11 = 1) ;Wait until Part Ready Signal Comes
X100.0               ;If Ready, Go there
RESET 01             ;Grasp it.
```

```
F250                  ;With Slow Speed
X50.0Y90.0            ;Move to Destination
DWELL 500
SET P02 = P02 + 1     ;Increment Time Count
GOTO 00 IF(P02>20)    ;If more than 10 sec. elapsed, Jump.
GOTO-03 IF(M12 = 1)   ;See if Box Ready
SET 01                ;Release the Gripper
SET P03 = P03 + 1     ;Increment Job Count
NEX
F1000
X25.0Y0.0             ;Go Back to Position Zero
SET 02                ;Job Done Signal
END
LOO                   ;Subroutine 00  Intermittent Bell
FOR 0                 ;Do forever
SET 02                ;ON 1 sec.
DWE1000
RESET 02              ;OFF 1 sec.
DWE1000
NEX
RET
```

[Example 7.3] Serpentine Pattern

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



```
A0z100        ;Address card 0, clear buffer
F324          ;Feedrate
HOM           ;Go home
```

```
t150          ;Set accel/decel for 150 msec.
LIN           ;Linear mode
X20000Y20000;Move X and Y to 20000 counts
DWE1024       ;Wait for 1 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         ;Incremental move X to 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
DWE100        ;Wait 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

L00
CIR1          ;Subroutine 00, circular CCW mode.
R=1187        ;Set radius to 1187 counts.
V2374         ;Incremental move one diameter
LIN           ;Linear mode
U-9500        ;Incremental move X back 9500 counts
RET           ;Return to main program

L01
CIR0          ;Subroutine 01, circular CW mode
R=1187        ;Set radius to 1187 counts
V2374         ;Incremental move one diameter
LIN           ;Linear mode
U9500         ;Incremental move X to 9500 counts
RET           ;Return to main program
```

[Example 7.4] Thumbwheel Switch Application

Thumbwheel switch outputs are connected to SMCC's parallel port (Address FFB8H). Since only two Thumbwheel switch outputs (BCD format) can be read at a time, connection must have multiplexing capability if more than two digits are desired. SEL0 - SEL7 (Address FFB9H) connections can be used for multiplexing purposes.

If thumbwheel switches with Diodes connected in series with each BCD output are used, no additional components are necessary in connecting up to 16 Digits of switches. Fig.7.4.1 illustrates Thumbwheel switch connection without any additional circuitry.

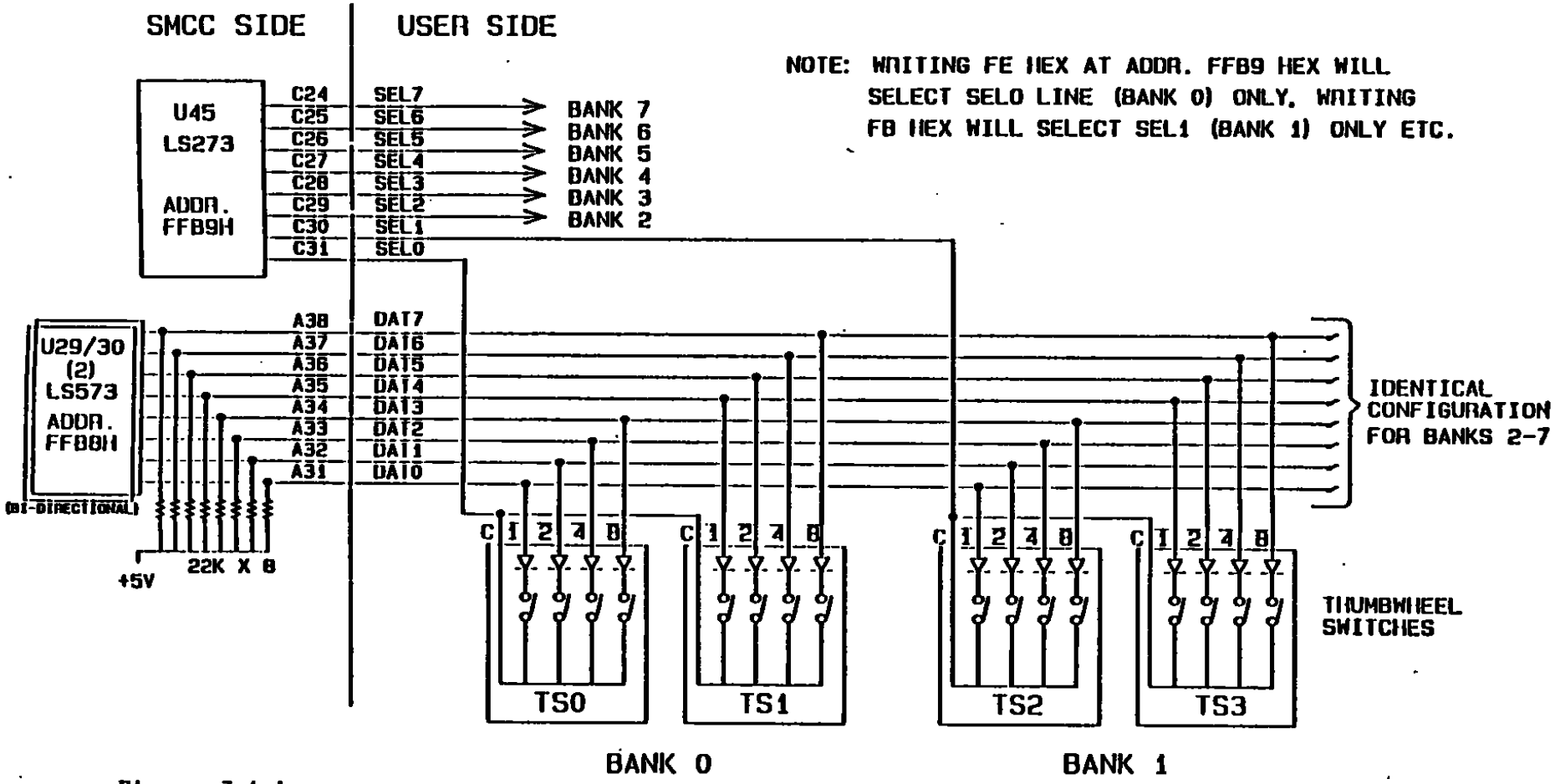

SMCC SIDE
USER SIDE
NOTE: WRITING FE HEX AT ADDR. FFB9 HEX WILL SELECT SEL0 LINE (BANK 0) ONLY. WRITING FB HEX WILL SELECT SEL1 (BANK 1) ONLY ETC.
U45
LS273
ADDR.
FFB9H
C24
C25
C26
C27
C28
C29
C30
C31
SEL7
SEL6
SEL5
SEL4
SEL3
SEL2
SEL1
SEL0
BANK 7
BANK 6
BANK 5
BANK 4
BANK 3
BANK 2
U29/30
(2)
LS573
ADDR.
FFB8H
(BI-DIRECTIONAL)
A38
A37
A36
A35
A34
A33
A32
A31
DAT7
DAT6
DAT5
DAT4
DAT3
DAT2
DAT1
DAT0
+5V
22K X 8
IDENTICAL CONFIGURATION FOR BANKS 2-7
C 1 2 4 8
TS0
TS1
TS2
TS3
THUMBWHEEL SWITCHES
BANK 0
BANK 1

Figure 7.4.1

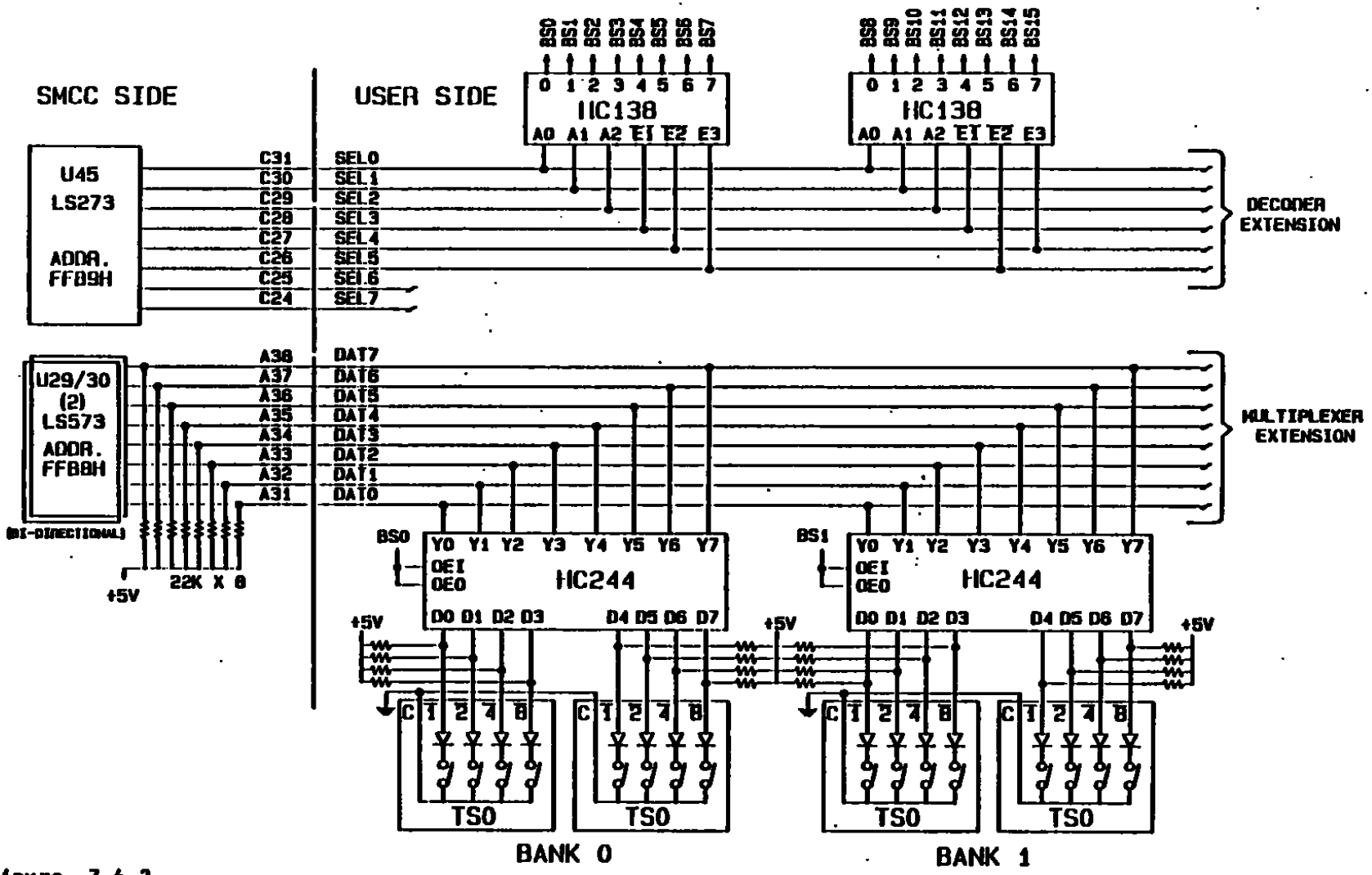

SMCC SIDE
USER SIDE
U45
LS273
ADDR.
FFB9H
C31
C30
C29
C28
C27
C26
C25
C24
SEL0
SEL1
SEL2
SEL3
SEL4
SEL5
SEL6
SEL7
BS0 BS1 BS2 BS3 BS4 BS5 BS6 BS7
BS8 BS9 BS10 BS11 BS12 BS13 BS14 BS15
0 1 2 3 4 5 6 7
HC138
A0 A1 A2 EI E2 E3
DECODER EXTENSION
U29/30
(2)
LS573
ADDR.
FFB8H
(BI-DIRECTIONAL)
A38
A37
A36
A35
A34
A33
A32
A31
DAT7
DAT6
DAT5
DAT4
DAT3
DAT2
DAT1
DAT0
MULTIPLEXER EXTENSION
+5V
22K X 8
BS0
BS1
Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
OEI
OE0
HC244
D0 D1 D2 D3 D4 D5 D6 D7
C 1 2 4 8
TS0
BANK 0
BANK 1

Figure 7.4.2

If more than 16 digits are desired, connection can be made with additional circuitry illustrated in Fig. 7.4.2. Depending on the application needs, user may design different decoders, or multiplexer circuits. Up to 512 Thumbwheel switches can be hooked-up with necessary decoders and multiplexers.
When Thumbwheel switch connection is made, proper program should be written to sequentially read switches, process data and convert to variables. Example 7.4a, 7.4b give you hints on writing Thumbwheel switch process routines for your own application.
Once Thumbwheel switches are connected, and programs are written, operator can set parameters and press switches to run or stop without using any terminals.

[Example 7.4a] Speed (RPM) Control Using Thumbwheel Switch

Assume the motor encoder has 500 lines and use X4 multiplier, one revolution equals 2000 counts. Attaches three Thumbwheel switches (TS0, TS1, and TS2) to the circuits given in figure 7.4.2 (use SEL0 and SEL1).
Writes data byte 00 to SMCC's output port at memory address ffb9 will select TS0 and TS1, writes data byte 01 will select TS2, Rotates Thumbwheel to get required RPM.

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

By fetching the data byte at SMCC's memory location ffb8, you get the setting from currently selected Thumbwheel switches. Since the initialization parameter i06 controls maximum reference speed, as long as you assign 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 (Chapter 5), The latter case will be executed only when an instruction that sends data to 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
```

```
A0z                 ;purge buffer
SET P06 = 33        ;conversion factor
SET P05 = 10        ;a multiplier
SET P07 = 0         ;a temporary variable, it will prevent
                    ;motor running at an intermittent value.
F1024               ;Maximum feed rate
FOR 0               ;endless loop
GOSUB 00            ;subroutine to read RPM value
I06(P07)            ;SET RPM READ FROM THUMBWHEEL SWITCHES
X20000Y20000        ;Demonstration motions
DWELL 128
X0Y0
DWELL 128
NEXT
END
L00                 ;SUBROUTINE 00
```

```
SET C02 = ffb8          ;assign input port for Thumbwheel data
mmffb9, 00              ;select TS0 (low nibble) and TS1 (high
                        ;nibble)
DWELL 50                ;set some delay
GOSUB 01                ;get low nibble, P10 contains the nibble
SET P02=(P10 * P05)     ;move one decimal position
GOSUB 02                ;get high nibble, P10 contains the
                        ;nibble.
SET P02=(P02 + P10)     ;now, we get first digit and second
                        ;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 low nibble, this is the last digit.
SET P02=(P02 + P10)     ;add last digit to hundreds and tenths
SET P02=(P02 * P06)     ;scale to counts/seconds.
RET
SET P07=(P02)
L01                     ;SUBROUTINE 01: Get low nibble
CASE 02, 10L            ;10 selections
SET P10 = 0
SET P10 = 1
SET P10 = 2
SET P10 = 3
SET P10 = 4
SET P10 = 5
SET P10 = 6
SET P10 = 7
SET P10 = 8
SET P10 = 9
RET
L02                     ;SUBROUTINE 02: Get high nibble
CASE 02, 10H
SET P10 = 0
SET P10 = 1
SET P10 = 2
SET P10 = 3
SET P10 = 4
SET P10 = 5
SET P10 = 6
SET P10 = 7
SET P10 = 8
SET P10 = 9
RET
```

[Example 7.4b] Speed and Job Number Control Using Thumbwheel Switch

Suppose a SMCC system is installed with 250 Line Encoder, and 30 Motor Revolution per

Inch Linear transducer. One inch end-effecter motion will then correspond to 30,000 Counts (x4 multiplication factor).
Variables vv, pp, and nn, which are to be specified by the operator 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 to 50.000 inches of end-effecter position.

nn: Number of Times. Range from 00 to 99.

B. Motion Sequence

Start from Position Zero, Do 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 with 25% Speed of vv.
4. Wait One Second
5. Wait Until M11 Turns ON, and Give pulse to M01.

C. Preliminary Steps

SMCC Variables are defined as follows, assuming that i38 (positional unit) is set to 1:1:0

P01 = vv (Range: 00 - 60)

P04 = 500 Multiplier to convert vv into Counts per sec. unit. 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 unit. 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 Inch, and 50 Times of Task operation.

```
TS1   TS0      TS2    TS5   TS4   TS7   TS6      TS9   TS8
 2     0        1     5.    9     2     5        5     0
 Bank0          Bank1 Bank2       Bank3          Bank4
-Speed-        ----------Position----------     -No.Job-
```

Note:

1. For Convenience, Decimal point is 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.

Main program calls a subroutine to read and process Thumbwheel switches. Subroutines L10

and L11 read one or two digits of thumbwheeel switches, and these data are processed in Subroutine L00.

```
                        ;------ Main Program
SET P04 = 500           ;Constants
SET P05 = 30
GOSUB 00                ;Read Thumbwheel Switches
I06 (P01*P04)           ;Speed Spec.
FOR (P03)               ;Do nn times.  P03 specifies nn
F1024                   ;Max. velocity
X(P02*P05)              ;Move to position specified by P02*P05
DWE1024                 ;Pause 1 sec.
F256                    ;25% Speed
X0                      ;Return
DWE1024                 ;Pause 1 sec.
GOTO-01 IF (M11=0)      ;Wait Until M11 is ON
SET 01                  ;Give Pulse to M01
RESET 01
NEXT
END
                        ;Subroutine to Process Thumbwheel   SW.
L00                     ;
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)
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 Thumbwheel Switch
                        ;and store in P10
L10
DWELL 50                ;Set some delay
SET C02 = ffb8          ;Read Parallel Port
SET P10 = 0             ;Reset P10
CASE 02, 10L
SET P10 = 0
```

```
SET P10 = 1
SET P10 = 2
SET P10 = 3
SET P10 = 4
SET P10 = 5
SET P10 = 6
SET P10 = 7
SET P10 = 8
SET P10 = 9
RET
                        ;L11 Reads Two digit Thumbwheel
                        ;Switches and store in P10
L11
DWELL 50                ;Set some delay
SET C02 = ffb8          ;Read Parallel Port
SET P10 = 0             ;Reset P10
CASE 02, 10L            ;Read Low Nibble
SET P10 = 0
SET P10 = 1
SET P10 = 2
SET P10 = 3
SET P10 = 4
SET P10 = 5
SET P10 = 6
SET P10 = 7
SET P10 = 8
SET P10 = 9
                        ;Read High Nibble
CASE 02, 10H
SET P10 = P10 +  0
SET P10 = P10 + 10
SET P10 = P10 + 20
SET P10 = P10 + 30
SET P10 = P10 + 40
SET P10 = P10 + 50
SET P10 = P10 + 60
SET P10 = P10 + 70
SET P10 = P10 + 80
SET P10 = P10 + 90
RET
```

To run the program, user first sets variables with Thumbwheel Switches, and issue "R" command, or close "RUN" switch connection.

[Example 7.5] GLUE DISPENSER PROGRAM:

This program is used to put glue on 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.

```
A0z200          ;Clear buffer and allocate 200
                ;program lines in the buffer
I150            ;Enable Y axis
F1024           ;Set feedrate to 100% of i06 value
FOR4            ;Start loop 4 cycles
X34000Y10552    ;Move glue gun to starting position.
SET01           ;Turn on M01 output which starts
                ;dispensing glue
DWE256          ;Wait for 256 msec.
V750            ;Move Y axis incrementally
U320            ;Move X axis incrementally
U-2638V21360    ;Interpolated, simultaneous move
CIR1            ;Circular mode, CCW (arc setup)
R=2833          ;Radius
U-2925V2565     ;Arc destination, make the circular move
RESET01         ;Turn off glue
LIN             ;Linear mode, cancel circular mode
U-480           ;Wipe off glue tip by moving X axis
SET02           ;Turn on chuck (rotate chuck 90 degrees)
DWELL51         ;Wait for 50 msec.
RESET02         ;Turn off chuck (stop chuck)
X34000Y9732     ;Move back to starting position
DWE20           ;Wait for 20 msec.
NEXT            ;End of loop, do this 4 times then
X0Y0            ;move X and Y back to 0 position
END             ;End of program
A0PURGE         ;
IF M11=0        ;Check if Emergency Stop is on
THEN M01=0      ;True, turn off glue
AND M02=0       ;Turn off chuck
AND I03=0       ;Disable COMMUNICATIONS handshake
AND"Q"          ;Quit program
AND I15=1       ;Disable Y axis
AND"b"          ;Set pointer to top of program
AND"h"          ;Home X axis only
AND P3=1        ;Set flag 3
AND I03=1       ;Enable COMMUNICATIONS handshake
IF M13=0        ;Check if X home flag is on
AND P3>0        ;
THEN I03=0      ;True, disable COMMUNICATIONS handshake
AND "q"         ;Quit current move
AND I15=0       ;Enable Y axis
AND"h"          ;Home both X and Y axis
AND P3=0        ;Resets flag 3
AND I03=1       ;Enable COMMUNICATIONS handshake
IF M14=0        ;Check if index chuck is on
THEN M02=1      ;True, rotate 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 T0 is finished with countdown
THEN M02=0        ;True, turn off index chuck
AND P5=0          ;Reset flag 5
END               ;End of PLC program
```

[Example 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 from the main program.

```
GOS00             ;Go to Subroutine 00
.....             ;
.....             ; insert some statement
.....             ;
LOO               ;Subroutine OO (Label OO).
mffca,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 i26
                  ;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 C20 = 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 true, interrupts are
                  ;disabled.  To re-enable interrupts, use
                  ;mffca, 1.
T50               ;Time mode moves, 50 milliseconds per
                  ;block for all succeeding moves.
L01               ;Subroutine 01 (Label 01).
U1000             ;Make an incremental move of 1000
                  ;encoder counts.
GOT01 IF(M20=0)  ;Go to Label 01 if 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 trigger is
                  ;received.
P1=HX             ;If interrupt has occurred, then set
```

```
                    ;parameter P1 to be equal to the frozen,
                    ;captured absolute position value of X
                    ;at the time trigger occurred.
                    ;It is important to note that if the
                    ;interrupt occurs during the
                    ;approximately 10 to 20 millisecond
                    ;period preceding each move
                    ;(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 main
                    ;program.
```

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

```
    L01             ;Subroutine 01 (Label 01)
    GOT02IF(X>P5)   ;Go to Subroutine 02 if X position
                    ;is greater than Parameter P5.
    U1000           ;Otherwise, make an incremental
                    ;move of 1000 counts.
    GOT01IF(M20=0)  ;Loop on Subroutine 01 until
                    ;interrupt occurs.
    P1=HX           ;
    AX=(P1 + P2)    ;
    L02             ;Subroutine 02
    RET             ;Return to main program
```

[Example 7.7] Multiple SMCC Sequencing Method

```
    AAq             ;Ssometimes 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 on 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 buffer
A0GOS01             ;Start card 0
AAFOR0              ;Start loop
GOS20               ;Wait for 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
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
SET03
SET04
SET05
DWE25
RES03
RES04
RES05
RET
L02                 ;Start card A2
```

```
GOS16
SET02
SET04
SET05
DWE25
RES02
RES04
RES05
RET
L16
GOT21IF04*15     ;Wait for all lines to be 1
DWE20            ;Wait 20 msec
GOT16            ;Go back to L16
L20
CAS04,16L        ;Wait for 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 this
                 ;card do something is "0", which
                 ;makes the card execute the L21
                 ;routine (a simple return in this
                 ;case).
A0GOT21          ;Card A0, 15 code "0", go to L21
DWE20            ;         14 wait 20 msec
DWE20            ;         13 wait 20 msec
DWE20            ;         12 wait 20 msec
DWE20            ;         11 wait 20 msec
DWE20            ;         10 wait 20 msec
DWE20            ;         9  wait 20 msec
DWE20            ;         8  wait 20 msec
DWE20            ;         7  wait 20 msec
DWE20            ;         6  wait 20 msec
DWE20            ;         5  wait 20 msec
DWE20            ;         4  wait 20 msec
DWE20            ;         3  wait 20 msec
DWE20            ;         2  wait 20 msec
DWE20            ;         1  wait 20 msec
DWE20            ;         0  wait 20 msec
A1DWE20          ;Card A1, 15 wait 20 msec
GOT21            ;         14 code "1", go to L21
DWE20            ;         13 wait 20 msec
DWE20            ;         12 wait 20 msec
DWE20            ;         11 wait 20 msec
DWE20            ;         10 wait 20 msec
DWE20            ;         9  wait 20 msec
DWE20            ;         8  wait 20 msec
DWE20            ;         7  wait 20 msec
DWE20            ;         6  wait 20 msec
```

```
DWE20           ;          5  wait 20 msec
DWE20           ;          4  wait 20 msec
DWE20           ;          3  wait 20 msec
DWE20           ;          2  wait 20 msec
DWE20           ;          1  wait 20 msec
DWE20           ;          0  wait 20 msec
A2DWE20         ;         15  wait 20 msec
DWE20           ;         14  wait 20 msec
GOT21           ;         13  code "2", go to L21
DWE20           ;         12  wait 20 msec
DWE20           ;         11  wait 20 msec
DWE20           ;         10  wait 20 msec
DWE20           ;          9  wait 20 msec
DWE20           ;          8  wait 20 msec
DWE20           ;          7  wait 20 msec
DWE20           ;          6  wait 20 msec
DWE20           ;          5  wait 20 msec
DWE20           ;          4  wait 20 msec
DWE20           ;          3  wait 20 msec
DWE20           ;          2  wait 20 msec
DWE20           ;          1  wait 20 msec
DWE20           ;          0  wait 20 msec
AAGOT20         ;Go back to L
L21RET          ;Return
```

[Example 7.8] An Automated Encoder Line Counting Program

Count and display Encoder line number

Hardware Configuration:

1: Parabolic SMCC.
2. Connect X-axis encoder C channel output (C/) to X-axis User Flag input, this input pin can be accessed at Baldor's SMCC Backplane TB1 pin 14.
3. Servo Gain been properly tuned.

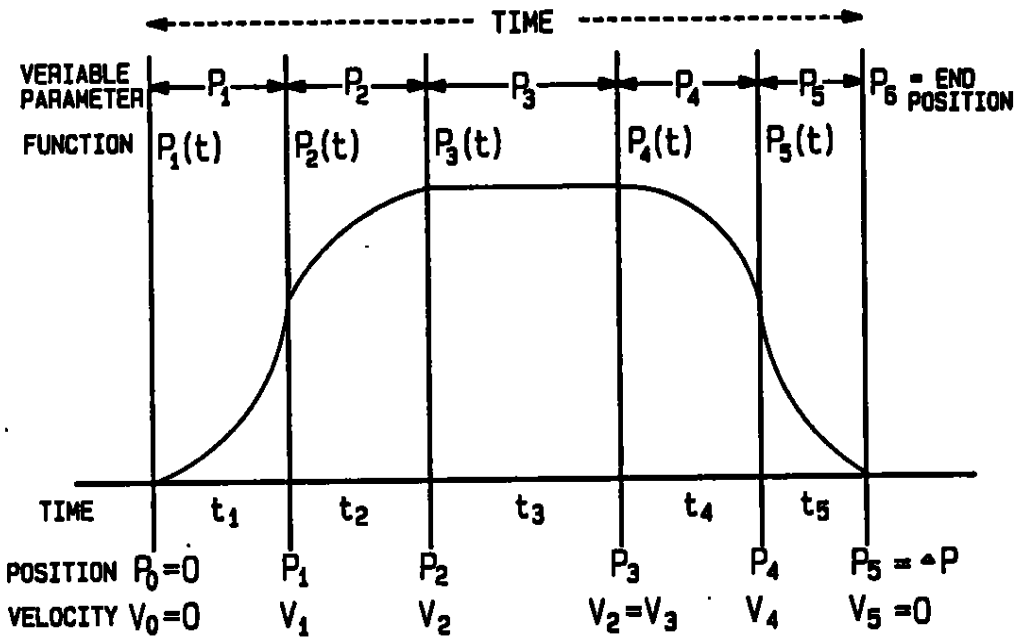
```
I393            ;or I397. multiplied by 4, to ensure stability.
I267            ;set User Flag Control bit to UF falling
F128            ;define Feed Rate, avoid low value (such as F32).
U5000^100       ;move 5000 counts until interrupt.  (triggered by
                ;encoder C/). Once interrupt being sensed, motor
                ;advances 100 counts.
                ;Do not use U5000^0.
                ;If no interrupt take place within 5000 counts,
                ;SMCC show error code "5".
                ;5000 is bigger than an ordinary encoder's
                ;line counts.
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 U5000^0 command so that a complete
              ;cycle is guaranteed.
DWE1024       ;allows stable servo.
SET P3=AX
SET P3=(P3-P2)
SET P4=4
SET P3=(P3/P4)
X=0
U(P3)         ;LCD's reading equals encoder count.
END           ;End of buffer program
```

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

Example 9: "S" Curve Accel And Decel (Requires Parabolic Option)

The 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 time from decel inflection point to end position.
P6 Defines total move distance.

The program is written so as to allow a completely generalized nonsymmetrical 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 = inflection point time value in binary milliseconds, P3 = run time value, and P6 = 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 machinery.

```
A0z100                ;Clear buffer
i061024               ;Set feedrate for lines/sec.
SETC20=febd           ;Set address for M20
SETM20=C20.5          ;Set M20 to save data for plot
SETP12=(P1+P2)        ;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
SETP13=(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)*(t1+2*t2+3*t3+2*t4+t5)
SETP7=(P6*P7/P12)     ;P7  = DP1=DP*t1*t1/(t1+t2)*
                      ;(t1+2*t2+3*t3+2*t4+t5)
SETP11=(P6*P11/P14);P11 = DP5=DP*t5*t5/(t4+t5)*
                      ;(t1+2*t2+3*t3+2*t4+t5)
SETP8 =(P6*P8/P13)  ;P8  = (DP1+DP2)=DP*(t1+2*t2)/
                      ;(t1+2*t2+3*t3+2*t4+t5)
SETP10=(P6*P10/P13);P10 = (DP4+DP5)=DP*(2*t4+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
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 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 run mode
SET20                ;Start 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
```

8: TECHNICAL INFORMATION

This chapter outlines some important features provided by 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 Parabolic SMCC much greater flexibility for blending of moves, smooth starting and stopping, and close control of the path of movement.

Even when executing lower-order trajectories, 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 (accel=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:

Directly, and

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:

```
t1024                ;specifies the time for the move
X10000:1024          ;specifies 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:

Vend = i06 * Vspecified / 1024 (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 deltaP/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 deltaP so that deltaP/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 deltaP 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 deltaP/t ratio.

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

```
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 deltaP/t ratio
t(P3)           ;constant automatically
U(P3*P4)
END
```

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

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

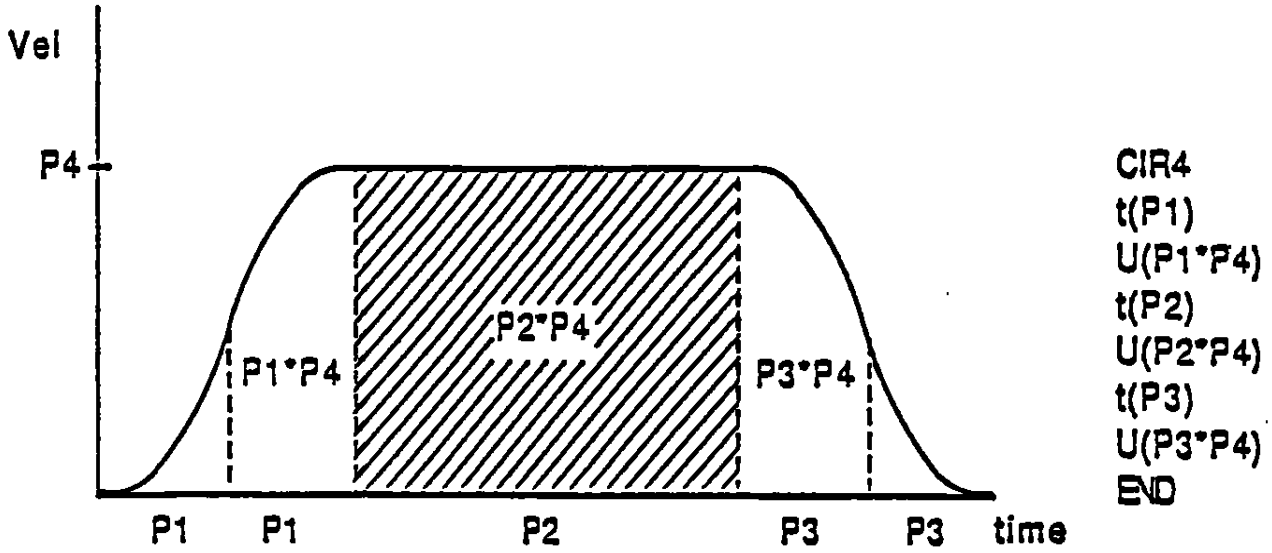


FIGURE 8.1 POINT-TO-POINT S-CURVE MOVE

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:

$$deltaP = (1/2) * 2t * V = V*t$$

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:

$$Apeak = 2 * V / (2t) = V/t$$

Now if we define:

```
Vmax = Maximum allowed velocity (cts/sec)
Amax = Maximum allowed acceleration (cts/sec/sec)
```

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

```
t = V/Apeak = Vmax/Amax (remember to convert to bin. msec.)
deltaP = V*t = Vmax*t
```

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

```
deltaP(slew) = TotalDistance - 2*deltaP(accel)
t(slew) = deltaP(slew)/Vmax
```

Example: Suppose we had a system with Vmax = 50,000 cts/sec and Amax = 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 deltaP = 50000 * 0.1 = 5000 counts.
Next, we compute the slew distance as 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:

```
t102            ;0.1 sec in binary milliseconds
U5000           ;acceleration distance
t307            ;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:

```
deltaP{min3part} = 2 * Vmax*t + 0.020*Vmax
                 = 2 * (Vmax^2)/Amax + 0.020*Vmax
```

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.

Minimum-time t = sqrt (deltaP/Amax)

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 Sec. The program would be:

```
t102        ;0.1 sec -- total move time = 0.3 sec.
U50000      ;incremental move total distance or
END         ;dwell -- something to 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):

```
3-point derived pos (n) = [r(n-1) + 4*r(n) + r(n+1)] / 6
3-point derived vel (n) = [r(n+1) - r(n-1)] / 2t
```

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

```
X=0             ;Define 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
```

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 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. 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: Send 'y' Command without any arguments just 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 completed, the host should send a 'y' Command without 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 need a host program (such as Monitor program) to translate the internal form (HEX table) into physical value.

The bytes number to be stored in defined servo cycles also implicitly defines the type of that data. This is further explained as follows:

Byte to collect in given servo cycles	data 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 bytes format is adopted, only first 4 byte and last 4 bytes is used to store X position and X Following error.
Velocity and Acceleration can be derived from positional data and given servo cycles

8.4 DMA Communications

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 command (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 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 Parabolic SMCC or be executed on-line by placing 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 AO 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 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 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 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 SMCC Setup for D.C motors. Certain amplifiers (such as Baldor's BTS-15 brushless amplifier) have built-in commutation function and act as 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 a Command "inn" without any parameter value is interpreted as 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]

i00         ;Query to check following error Fault Limit
            ;SMCC will respond with currently stored values.
i00 2000    ;New parameter value replaces old value.
i002000     ;Same as above.
I00 2000    ;Buffer command - will be processed at the time of
            ;program execution.
```

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

---

i00 Set following error Limit (FAULT)
Range (0 +/- 32767), Default (2000), Unit: Encoder Count

SMCC keeps track of following error (Position error between the desired position and 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 amplifier 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 in hardware to resume operation. The FEOD/ output may be used to turn system power off.
If a negative limit is used in i00, the following error is still detected and the amplifier 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 following error Fault, set i00 to 0.

---

i01 Set In-Position Band
Range (0 - 65535), Default (100) Unit: Encoder Count

When SMCC is at rest. it turns on the in-position Light and IPOS/ Output whenever the actual position is within the specified counts from its current 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 +/- 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 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 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 Handshake Enable in Host Communication
Range (0,3), Default (1), Unit: None

0: Handshake is disabled. Processor does not send handshake messages. But will respond upon reception of query commands.

1: Handshake is enabled. Processor sends <LF> character upon reception of <CR>, and sends "BELL" upon Unrecognizable commands.

2: Handshake is enabled; responds with <ACK> for valid commands or <BELL> for illegal commands. Messages are sent without <LF>.
Data is sent as: DATA, <CR>, (DATA, <CR>, . . .) <ACK> (with <ACK> sent after last line only).

3: Handshake is enabled; sends <ACK> or <BELL> as appropriate. Data is preceded by <LF> and followed by <ACK>. Data is sent as: <LF>, DATA, <CR>, (<LF>, DATA, <CR>, . . .) with <ACK> (after 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 Home Feedrate
Range (0 - 255), Default (0), Unit: Encoder counts/second.

This parameter sets up the speed of homing operation with respect to reference speed set by

i06. Home feedrate is calculated by the following equation.

Home feedrate = (i04 / 256) * i06

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

---

i05 Set Servo Time Period
Range (0 - 65535), Default (487), Unit: Microsecond

This parameter specifies period of servo control and is calculated by:

Servo Time Period = 2 * (i05 + 1) 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 in normal condition. If servo time period is shortened too much, there will be insufficient time to handle communications and PLC and cause the SMCC hang up.

---

i06 Set Reference (Maximum) Feedrate
Range (0 - 8388607), Default (2000), Unit: EC/sec.

This parameter specifies Reference Feedrate so that 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 "F" (Feedrate) Command:

Feedrate = (nn / 1024) * i06 (Encoder counts/Sec.) and
Jog speed = (i31[i51]/256) * i06
Home Feedrate = (i04/256) * i06

Following conditions will be achieved:

```
100% Feedrate..................... when enter F1024 command.
Maximum Home Feedrate........ when i04 set as 255
Maximum Jog rate.............. when i31[51] set as 255
```

---

i07 Set Time Scale
Range (0 - 65535), Default (65498), Unit: None

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

i07 = 65536 * (2 * (i05 + 1)/dT)
where dT = Units for 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 controls the time base for 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 Acceleration/Deceleration Time
Range (0 - 65535), Default (512), Unit: 1/1024 sec.

This parameter governs total accel/decel time whenever speed change occurs. When i08 = 1024, accel/decel time is one second.
SMCC uses time-specified accel/decel format instead of rate specified format. 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. Accel and decel times are equal and both controlled simultaneously. It is possible to make 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 a H (feed hold) Command on a jog move.

i08 is read once upon power turn-on 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 Position Integration Mode
Range (0,1), Default (1), Unit: None

To reduce following error, SMCC utilizes 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 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 Limit for 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 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, 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. Deceleration rate is controlled by i08 or % slew rate if in a calculated move. 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 zero value (stop) is reached. 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 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 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 Motor Type
Range (0), Default (0) Unit: None

Set 0 to select Brush type DC Motor Control - Two axis control is possible. (See i15)

The SMCC can be configured such that it commutates the 3-phase motor current of a brushless 3-phase drive. In this case, set i13 1. This is the proper setting for the SBTS 10 and SBTS 20 drives which both utilize Micro-SMCC internally.
All other Baldor DC and AC drives where the commutation sequencing is done at the motor or brushless drives that commutate without SMCC, In this case, set i13 0. The drives include: LIM, SUM, THM, TSD, BTS 15, BTS 10, BTS 20 and SBTS 15.

i14 Select 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 X axis) and ANA3 (pin J1-A1 for Y axis)] are controlled by i14 as follows:
Note that regardless of the mode selected, ANA0 and ANA1 channels are activated to perform overload protection.

0: A/D converter Mode Disabled. Analog inputs ANA2 and ANA3 are not used. (Default)

1: Feed Pot Mode (same as % Command) - SMCC uses ANA2 input to control feedrate (ANA3 is not used), allows a simple variable voltage input (0 - +10V DC) through a potentiometer to be used as program execution rate control.
This is just like rate controlled by % 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, as noise on the line translates into very noticeable movement jitter.
Do not have i16 = 2 when i14 = 1, because in such 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 jog moves: ANA2 controls X-axis speed; and ANA3 controls 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 variable voltage input (+/- 10V) through a Potentiometer, and is bidirectional with 4 speed ranges controlled by the XJPL (X axis Jog Positive, J1-B30), and XJMI (X axis Jog Negative, J1-32) [J1-B39 and J1-B31 for Y axis] inputs. This will increase relative resolution for low speed operation.

JogPositiveInput	JogNegativeInput	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 motor current and provide I^2t protection.

---

i15 Set Y Axis Disable in DC Motor Mode (i13=0)
Range (0,1), Default (1), Unit: None

0: Y axis is enabled.

1: Y axis is disabled.

If only one axis DC Motor is to be controlled, connect it to "X" axis, and configure i15 as "0" to block "Y" axis logic process. If Y axis is disabled, the "Y" error output can be used as a programmable voltage generator by using the OYn Command.

NOTE: In a one-axis only condition, the limit switches for Y axis MUST be disabled (grounded); otherwise SMCC will not make any moves.

---

i16 Select Handwheel Operating Mode
Range (0,1,2,3), Default(0), Unit: None

The ratio of handwheel to motor movement or scale factor is governed by i27[i47] and the mode of operation by i28[i48].

1: Standard Mode -- Moves strictly follow handwheel encoder input. No calculated moves are possible in this mode. This is a special mode allowing very fast servo cycle thus permitting "tight" handwheel following. Scaling is governed by i27[i47] and Handwheel Mode Control is governed by i28[i48].

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

Time Base = Input frequency x Mult.Factor x ScaleFactor

An 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 X handwheel; otherwise same as Mode 2.
NOTES: When using i16 Mode 2 or Mode 3, it is a good practice to disable the X and Y by setting i28[i48] to 3.
When set i16 in 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 Velocity Display Time Base
Range (1 - 2 *31), Default (30000), Unit microsecond

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

i17 ="sample period" in microseconds

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

```
Velocity  =   Delta position during sample period
          =   counts/30 milliseconds
          =   1/2000 Rev Per 30 milliseconds
          =   Rev/60 seconds
          =   RPM
```

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

---

i18 Select Display Type to be Used
Range (0-3), Default (3)

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

0 is LED display - 2 units x 1 line (8 digits/line)
1st display shows X position.
2nd display shows Y position if i13 = 0;

1 is LCD display - 2 units x 2 line (24 digits/line)
1st display shows X position and X velocity on 1st line; X F.E. and X status on 2nd line.
2nd display shows Y position and Y velocity on 1st line; Y F.E. and Y status on 2nd line.

2 is LCD display - 2 units x 1 line (40 digits/line)
1st display shows X position, X velocity, X F.E., and X status;
2nd display shows Y position, Y velocity, Y F.E., and Y status.

3 is LCD display - 1 unit x 2 lines (40 digits/line)
1st display shows X position, X velocity, X F.E., and X status on 1st line; Y position, Y velocity, Y F.E., and Y status on 2nd line.
2nd display can show X F.E. and X status on 1st line; Y F.E. and Y status on 2nd line; useful if actual displays are shorter than 40 digits long.

X position format is governed by i38.
Y position format is governed by i58.
X velocity format is governed by i17 and i38.
Y velocity format is governed by i17 and i58.
X following error format is governed by i38.
Y following error format is governed by i58.
X status shows bits of 3rd status byte, feb7 for Parabolic SMCC)
Y status shows bits of 4th status byte. feb8 for Parabolic SMCC)

Refer to the SMCC Memory Bit Map for definitions of the individual bits.
If you are not using a display setting, assign 0 to i18 will shrink background time and

cut interruption to other background tasks.

---

i19 Set 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 characters <CNTRL B, C, or D>.

---

i20[i40] Set Proportional Gain Constant
Range (0 - 65535), Default (50), Unit: None

This is the major gain constant in the software oriented PID controller. The SMCC output a error correction signal which increase, in direct proportion, the amount of following error been observed. The output is as follows:

(1.25 * i20[i40])/65536 = VOLTS/ENCODER COUNT

Usually i20 is proportional to the "stiffness" of the loop.
The SMCC's servo algorithm is shown here for reference.

```
Eout =          i20*k0*{EP(n) - [i21*AV(n)] + i22*DV(n)
                +i30*k1*[DV(n)-DV(n-1)] + i23*k2*IP(n)}
Where:
i20 =           Proportional Gain
k0 =            1.25 volts/65536
                (Used to scale 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 accel feedforward)
i23 =           Integral gain
k2 =            1/4096 (Used to scale integral gain)

                n - 1
IP(n) =         SUM EP(j) (integral of 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 Differential Gain Constant (Feedback Term)
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 used, 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] Velocity Feedforward Gain
Range (0 - 255), Default (0), Unit: None

i22 is used to generates a term proportional to the desired velocity that is added into the servo command. This term is very useful in reducing 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 system stability. Also, it is more useful than error integration for reducing following errors on variable-speed moves, because error integration cannot keep up with varying speeds as feedforward does.
In a system without a tachometer, i22 should generally be set equal to i21.

---

i23[i43] Set Integral Gain Constant
Range (0 - 255), Default (0), Unit: None

The integral gain of the SMCC is used to correct residual position errors due to off-balance 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 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 only when SMCC is at rest; when i09 = 0, i23[i43] is effective at all times.

---

i24[i44] Select Home Direction
Range (0,1), Default (0)

0: Home in negative direction.
1: Home in 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 Home Offset
Range (0 - +/-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[45]. (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 homing procedure.

---

i26[i46] Set Home Flag and User Flag Control Bits
Range (0 - 7), Default (0), Unit: None

SMCC can receive interrupts from several different sources based on hardware configuration. i26 configures this function so that interrupt jumps 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[46] 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 X-axis inputs that generate the trigger in the "Go Until" move (Xd ^ d). The X-axis interrupt acts on both axes for this situation only. i46 establishes the condition of Y-axis inputs that generate the trigger for WUT (Wait Until Trigger) moves.

---

i27[i47] Set Handwheel Scale Factor
Range (0 - 65535), Default (4096), Unit: None

This determines the ratio of handwheel count vs. motor encoder count. Desired position indicated by handwheel is calculated as follows.

Desired Position = (i27[i47] / 4096) * Handwheel Count

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

i27 = 4096 sets one-to-one match between two values.

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

The external frequency is used by SMCC to calculate the effective "%" it will use when it makes its moves. Calculations are made every servo cycle and "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 a input frequency that will not cause the % command to overflow (Maximum value for % command is 65535, When overflow occur, 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 so that a T of 1024 represents 1 second of time. If % is changed to 1/2 of its value, 32749, then a T of 1024 will represent a time of 2 seconds. The % Command therefore controls the time base of 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: Y axis is expected to vary its pitch every 10th rotation of the spindle. Total travel is 10 inches; 10 rotations of spindle should cause traverse to move 1 inch. X axis moves across at constant pitch. Both X and Y are "slaved" to the spindle axis and the spindle axis encoder is fed to the SMCC's "X Handwheel" input and used to establish the "%" value which controls the SMCC's time base.

```
A0              ;
z100            ;clear buffer
i283            ;disable X and Y handwheel mode
i483            ;
                ;assume 8000 counts/rev. of spindle.
i163            ;X axis handwheel is time base for SMCC
i27100          ;Make 1 binary mSecs. = 100 counts
i4732768        ;100% feedrate (%) = counts/mSecs.
                ;=(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
                ;Actual time will be 1 Sec. per 10 revs. 600RPM
                ;total move, 10 seconds, 80000 counts.
T800            ;Time for 10 revolutions of spindle
U(P1)V(P2)      ;10 revs. P1 and P2 are 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 axis is stopped before switching from
                ;i163 to i160. On the fly switching of time base
                ;is possible; just take care that a big
                ;velocity change will not occur.
END             ;End of program
                ;P1= wire traversed each 10 revs of spindle
                ;in encoder counts for X axis
                ;P2, 3,4,5,6 same for Y axis
                ;Just send new variable parameter values to
                ;change the program.
                ;All P variables represent distance to move
                ;the traverse axis in encoder counts.
```

---

i28[i48] Select Handwheel Mode Control
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 - Machine zero position is not changed or translated. Position follows handwheel move and position display changes - inactive during a calculated move.

1: Position Offset Mode - Machine zero position is changed and is translated as the handwheel moves. Position display does not change, active during run or stop.

2: 1/T Mode -- 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 i27[i47] description.

3: Handwheel position control disabled but the specified handwheel signal source (external or other axis) may be used for time synchronization (see i16).

---

i29[i49] Set Handwheel Encoder Control Bits
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.

```
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 the handwheel inputs are enabled (see i28[i48]).

---

i30[i50] Acceleration - Feedforward Gain
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 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 Jog Feedrate
Range (0 - 255), Default (200) Unit: None

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

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

Note that Jog Feedrate is affected by both i06 and i31[i51]

Jog speed also can be changed by using "%" on-line or "POT" in the Buffer Command. This, however, will also affect the time values used in the program.

---

i32[i52] Set Normal PWM Limit Control
Range (0 - 255), Default (128) Unit: None

Limits error output to amplifier. 0 is no output, 255 is full output or +/-10 volts.

---

i33[i53] Set Protective PWM Limit Control
Range (0 - 255), Default (64), Unit: None
Sampling Rate: 43 msec

Suggested Setup: i32[i52] > 2 x i33[i53]

This parameter sets up maximum allowable PWM limit. Once 75% of i32[i52] is reached and maintained for approximately three seconds, SMCC goes into Protective Mode which uses Protective PWM Limit Value (i33[i53]) until current goes down to corresponding value. It then restores the output limit to i32[i52] level. The I^2t output and LED indicator are turned on during protective cycle.

---

i34[i54] Set Backlash Compensation
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 direction is changed, 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 were set to 100 counts, then a X2000 move would actually cause a move of 2100 encoder counts. If the next following command were 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 been observed when a move starts in a new direction can 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 Software Positive Position Limit
Range (-1 to [2*27]-1),
Default (-1), Unit: Encoder Count

i35[i55] are used to prevent beyond set limits. Value of -1 sets the limits to 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 Software Negative Position Limit
Range (-1 to 67,000,000),
Default (-1), Unit: Encoder Count

i36 serves the same purpose as i35, A -1 value sets limits to 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 Position Range
Range (+/- [2*27]-1),
Default (0), Unit: Encoder Count

If a negative number is specified, range is set up for both directions up to the specified absolute value. If a positive number is specified, range is set up for positive direction only (1 - specified range). Set to zero to disable this function.

---

i38[i58] Set Position and Display Scale and Format

```
(n:m.d), Default (1:1.0)
Range:
n    (-2*31 - +2*31 - 1)
m    (-2*31 - +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]

Motor has 500 lines/Rev. encoder. Lead screw is 5 pitch.
It is desired to display position in inches with 2 decimal points.

1) Multiply 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 x 2000 = 10000 encoder counts/inch.
4) Set i38[i58] so n=10000 (convert to inches), m=100 (show hundredths of an inch), and d=2 (decimal point two places in): 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 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 Position Encoder Mode Control Bits

Range (0-7), Default (3)

```
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 as 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 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 encoder input (See i29[i49] for controlling the operating mode of the handwheel encoder.)

---

i60[i61] Deadband
Range (0-255), Default (0), Unit: None

Specifies a +/- position range from desired position where SMCC does not close the servo loop. This is useful in systems that have a lot of backlash between the motor and the load.

---

i62 PLC Enable
Range (0, 1), Default (0), Unit: None

This parameter is used to activate or deactivate the PLC program stored in SMCC. Set it to 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.

---

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 for parabolic version, 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 version 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 distance slightly greater than twice the radius to be done without error (this situation usually occurs due to roundoff 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 Happy)
1 = 1 Reserved
2 = Following Error Failure (i00)
3 = Not enough time to calculate move
4 = Circle radius too small for specified radius.
5 = "Go Until" command did not receive a trigger
6 = "Go Until" command did not receive a trigger
7 = Communications Error (Invalid Character)
8 = Amplifier Failure, (without Fault Data Bits)

code 9, A, B, C, D, E, F Reserved

B.2 SMCC STATUS BYTES

The SMCC send Status bytes upon the reception of "?" and carriage return ("?" is status query command).

Bit	Byte 1 (feb5)	Byte 2 (feb6)
0	Home Complete (Y)	Time Mode (not feedrate)
1	Home Complete (X)	Single Step
2	Continuous Move	Buffer Full
3	Block Request (MDI mode)	Parabolic Mode
4	Auto Divide	Limit Switch
5	Feed Hold	Not Ready (X)
6	Running (any move)	Not Ready (Y)
7	Timer Active	Not Ready For Move

Bit	Byte 3 (feb7)	Byte 4 (feb8)
0	X Jog Positive	Y Jog Positive
1	X Jog Negative	Y Jog Negative
2	X Limit Pos. (soft or hard)	Y Limit Pos. (soft or hard)

3	X Limit Neg. (soft or hard)	Y Limit Neg. (soft or hard)
4	X Follow. Err. (i10 exceeded)	Y Foll. Err. (i10 exceeded)
5	X I2T Limit (low true)	Y I2T Limit (low true)
6	X Not In Position	Y Not In Position
7	X Zero Velocity (commanded)	Y Zero Velocity (commanded)

Bit	Byte 5 (feb9)	Byte 6 (feba)
0	Edit Mode	Auto Divide Req
1	Insert Mode	Extended Move Flag
2	Extended Read	Extended Move Flag
3	Circle Mode	Home Has Been Done Flag
4	Wait (X)	CCW Circle
5	Wait (Y)	Circle > 180
6	Backlash (X)	Position Frozen (after ^V)
7	Backlash (Y)	Sub. Buffer Open

Bit	Byte 7(febb)	Byte 8 (febc)
0	Servo Pointer	Servo Pointer
1	Servo Pointer	Servo Pointer
2	Servo Pointer	Servo Pointer
3	Dwell Request	Servo Pointer
4	Home in Progress until capture	Servo Pointer
5	Dwell in Progress	Step Request
6	Continuous Move Request	Servo Pointer
7	Memory Request	Servo Pointer

Appendix C: On-Board DIP Switch

C.1 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

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

Address		Description
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^2T/
FF01	Bit 5	MAO1, 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.
		EQUL= 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	MAO2, Machine Output #2.
FFB4	Bit 7	MAO3, Machine Output #3.
FFB5	Bit 6	MAO4, Machine Output #4.
FFB5	Bit 7	MAO5, 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

SMCC automatically sets following definitions:

Address Definition	Machine Outputs	Machine Inputs
C01 = FF01	MO1 is M01 = C01.5	MI1 is M11 = C04.0
C02 = FFB4	MO2 is M02 = C02.6	MI2 is M12 = C04.1
C03 = FFB5	MO3 is M03 = C02.7	MI3 is M13 = C04.2
C04 = FFB7	MO4 is M04 = C03.6	MI4 is M14 = C04.3
C05 = FFBC	MO5 is M05 = C03.7	MI5 is M15 = C05.0
		MI6 is M16 = C05.1
		MI7 is M17 = C05.2

MINI INTERNAL RAM 10/19/88

PARABOLIC

```
                        ORG   0FE00H  ;INTERNAL RAM
FE00            RBUFF:  DS    32
        FE00    TBUFF   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            JCCEL1: DS    8       ;DESIRED ACCEL DOT X-AXIS
FE7C            JCCEL2: 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            HFDPOT: 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        ;I2T 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        ;I2T 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 ;INTERGATOR 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 ;HIGEST 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  ;CONTINUOS 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    I2T1    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    LMTP2   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    I2T2    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.;AUTO DIVIDE REQUEST BIT 0
        05D1    EXMOVE  EQU  XBITS+1.;EXTENDED MOVE FLAG (2 BITS)
        05D2    EXMOV1  EQU  XBITS+1.
        05D3    HMDONE  EQU  XBITS+1.  ;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   ;CONTINUOS MOVE REQUESTED
        05DF    MEMREQ  EQU  SFLAG.7   ;MEMORY MODIFY REQUESTED

FEBC            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    UMOVIE  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
FEDE            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
FEEE            IR14:   DS   1
FEEF            IR15:   DS   1
        FEE0    IRP0    EQU  IR0
        FEE2    IRP1    EQU  IR2
        FEE4    IRP2    EQU  IR4
        FEE6    IRP3    EQU  IR6
        FEE8    IRP4    EQU  IR8
        FEEA    IRP5    EQU  IR10
        FEEC    IRP6    EQU  IR12
        FEEE    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 05/09/89

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       ;CONTINUOS 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 L00-L63
A180            CTABLE: DS    128     ;SYMBOL TABLE C00-C63
A200            MTABLE: DS    64      ;SYMBOL TABLE M00-M63
A240            PTABLE: DS    60      ;PARAMETER TABLE P01-P15
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 TIMER
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
```

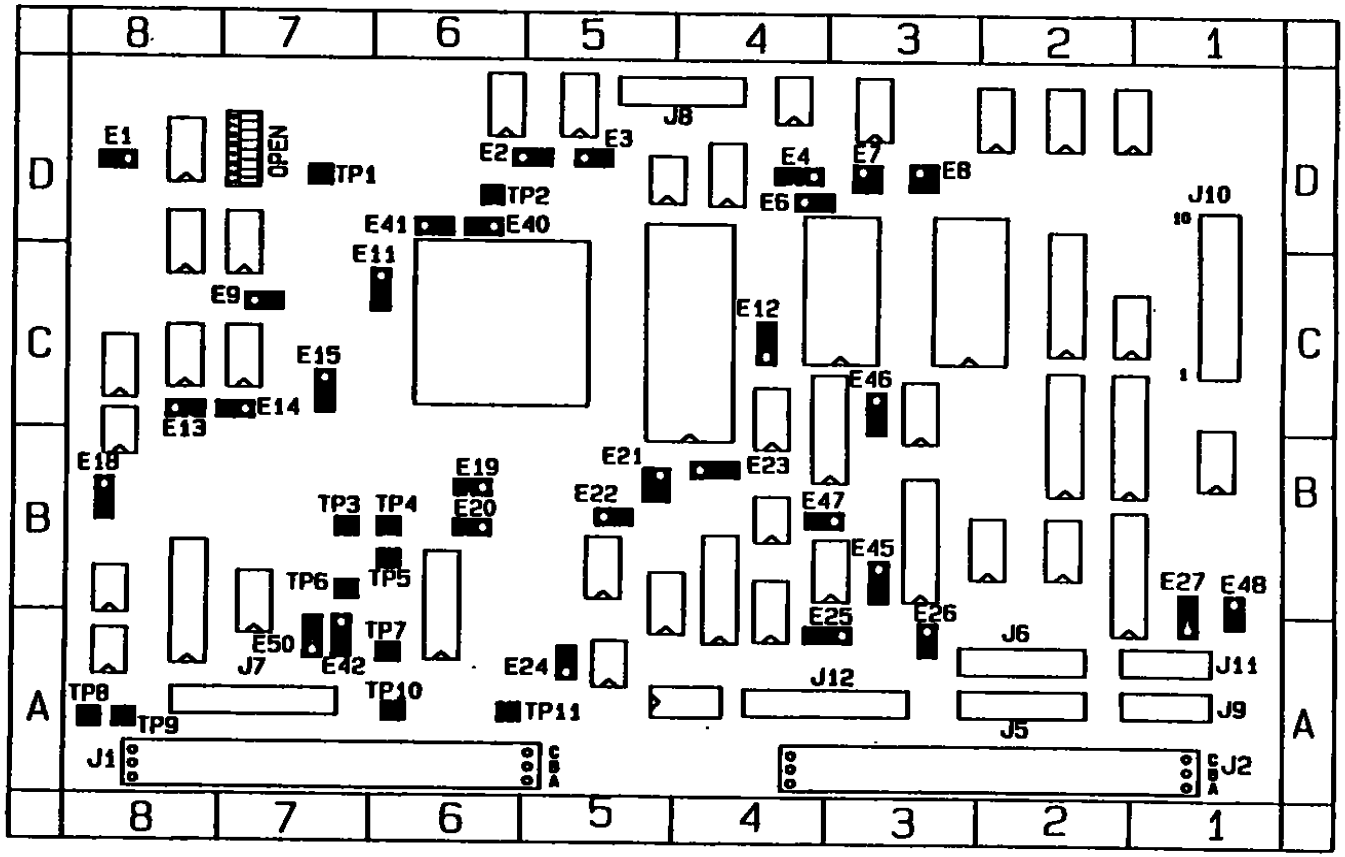
```
A359            HPOS2:   DS   4      ;HOME POSITION Y-AXIS
A35D            LSTMOD:  DS   1      ;EXTENTED READ MODE
A35E            VTIME:   DS   2      ;VELOCITY TIMER
A360            DSPOS1:  DS   4      ;DISPLAY POSITION X-AXIS
A364            DSPOS2:  DS   4      ;DISPLAY POSITION Y-AXIS
A368            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      ;EXTENTED READ COUNTER
A39E            LSTPTR:  DS   2      ;EXTENTED 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 ADD.
A402            I65:    DS   2      ;Y ENCODER/RESOLVER BASE ADD.
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
E2	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 AO.
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)

	E POINT	LOCATION
1.	E1	D8
2.	E2	D6
3.	E3	D5
4.	E4	D4
5.	E6	D3
6.	E7	D3
7.	E8	D3
8.	E9	C7
9.	E11	C6
10.	E12	C4
11.	E13	B8
12.	E14	B7
13.	E15	C7
14.	E18	B8
15.	E19	B6
16.	E20	B6
17.	E21	B5
18.	E23	B4
19.	E24	A5
20.	E25	A3
21.	E26	A3
22.	E27	A1
23.	E40	C6
24.	E41	C6
25.	E42	A6
26.	E45	B3
27.	E46	A3
28.	E47	B3
29.	E48	B1
30.	E50	A7
31.	TP1	D7
32.	TP2	D6
33.	TP3	B7
34.	TP4	B6
35.	TP5	A7
36.	TP6	A7
37.	TP7	A6
38.	TP8	A8
39.	TP9	A8
40.	TP10	A6
41	TP11	A5



SMCC E-POINT

E POINT	PHYSICAL LAYOUT	LOC.	DESCRIPTION	DEFAULT
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
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

E POINT	PHYSICAL LAYOUT	LOC.	DESCRIPTION	DEFAULT
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.	2-3Jumper installed card is configured for 8KX8 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
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 installed 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

E POINT	PHYSICAL LAYOUT	LOC.	DESCRIPTION	DEFAULT
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 B-4 B-4	JUMP Pin 1 to Pin 6 to send optically isolated BM01 to JMACH (J1) & JPWR (J11) JUMP Pin 2 to Pin 5 to send optically isolated BIPOS/ to go to JMACH (J1) & JPWR (J11) JUMP Pin 3 to Pin 4 to send optically isolated BIPOS to go to JMACH (J1) & JPWR (J11)	2-5 jumper installed IPOS/output
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)

E POINT	PHYSICAL LAYOUT	LOC.	DESCRIPTION	DEFAULT
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 (JPAR-RS232) CONN.	No jumper installed
E27	*3 *2 *1	A-1	JUMP Pin 1 to Pin 2 to connect A5 to R/W displays & JPAR/RS232.	2-3 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 installed (X encoder not 1/T)
E41	1 2 * *	C-6	Same as E40, but for Y axis.	1-2 installed (Y encoder not 1/T)
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 (X encoder non-diff.)
E43			NOT USED	
E44			NOT USED	

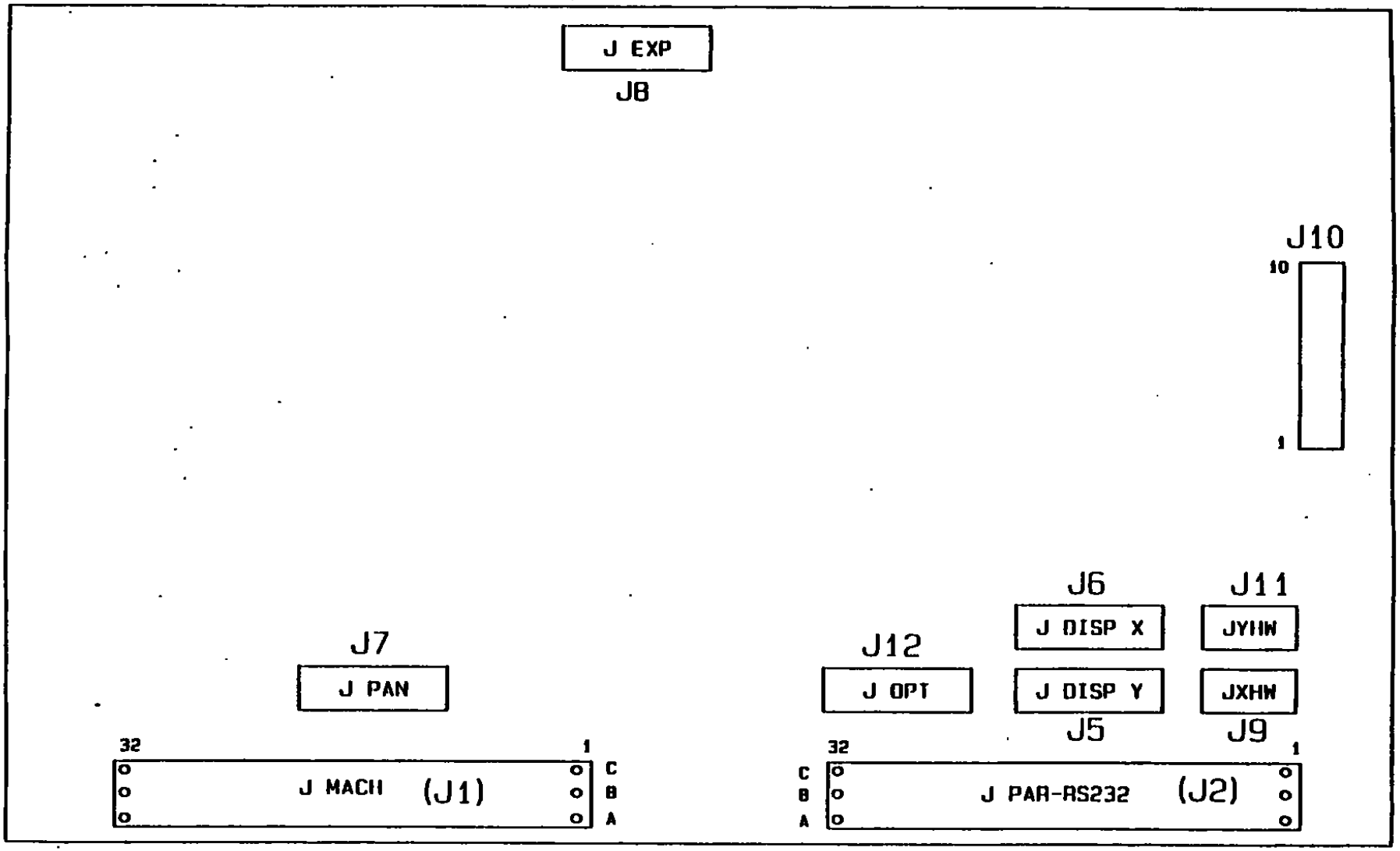
E POINT	PHYSICAL LAYOUT	LOC.	DESCRIPTION	DEFAULT
E45	*1 *2 *3	B-3	JUMP Pin 1 to Pin 2 to obtain differential X handwheel input mode. JUMP Pin 2 to Pin 3 to obtain non-differential X handwheel input mode, this will bias X-HW negative inputs to 1/2 VCC = 2.5V	2-3 (X - HW enc non-diff.)
E46	*1 *2 *3	C-3	Same as E45, but for Y handwheel.	2-3 (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

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 TP (X ENCODER CHANNEL C TP)
TP11	A-5	INIT/ RESET INPUT TP, LOW = RESET CONDITION HIGH= NON-RESET CONDITION

Apendix F SMCC Connectors and PIN Assignments

SMCC CONNECTORS	LIST OF MATTING CONNECTIONS
J1 (JMACH) -	AMLAN P/N C96F3LA+B+C
J2 (JPAR-RS232) -	AMLAN P/N C64F3SCHM.Z.A+C 3M P/N 6964-0000-EF OR equivalent
J3 -	NOT USED
J4 -	NOT USED
J5 (JDISPX) -	H and T P/N 4003-14-00-P5 3M P/N 3385-6600 or equivalent
J6 (JDISPY) -	H and T P/N 4003-14-00-P5 3M P/N 3385-6600 or equivalent
J7 (JPAN)	H and T P/N 4003-26-00-P5 3M P/N 3399-6600 or equivalent
J8 (JEXP) -	H and T P/N 4003-20-00-P5 3M P/N 3421-6600 or equivalent
J9 (JXHW) -	H and T P/N 4003-10-00 P5 3M P/N 3473-6600 or equivalent
J10 (POWER) -	NONE
J11 (JYHW) -	H and T P/N 4003-10-00-P5 3M P/N 3473-6600 or equivalent
J12 (JOPT) -	H and T P/N 4003-26-00-P5 3M P/N 3399-6600 or equivalent

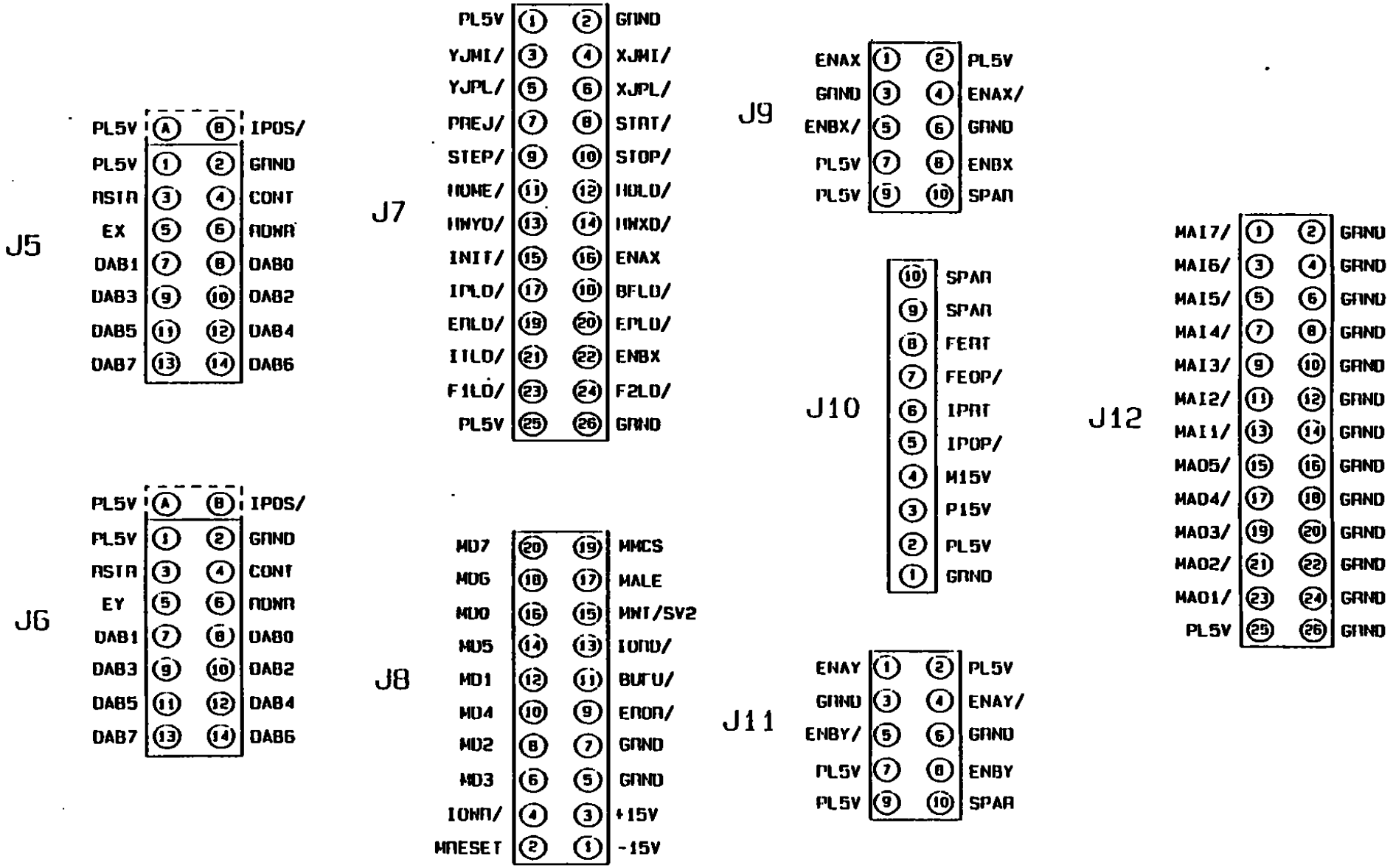

J EXP
J8
J10
10
1
J6
J11
J DISP X
JYHW
J7
J12
J PAN
J OPT
J DISP Y
JXHW
J5
J9
32
1
C
B
A
J MACH (J1)
J PAR-RS232 (J2)

J1

	A	B	C
32	FALT/	XJNI/	FALT/
31	AENA	YJNI/	AENA
30	AGND	XJPL/	AGND
29	P15A	YJPL/	P15A
28	MAI5/	STRT/	MAI5/
27	MAI6/	PREJ/	MAI6/
26	MAI7/	STOP/	MAI7/
25	M15A	STEP	M15A
24	PHA3	HOLD/	PHA1
23	PHA3/	HOME/	PHA1/
22	SPAR	INXD/	SPAR
21		INYD/	21
20		SPAR	20
19		INIT/	19
18		BFLD/	18
17	CHAY	IPLD/	17 CHAX
16	CHAY/	EPLD/	16 CHAX/
15	CHBY	ERLD/	15 CHBX
14	CHBY/	SPAR	14 CHBX/
13	CHCY	F1LD/	13 CHCX
12	CHCY/	F2LD/	12 CHCX/
11	YHFL/	F1LD/	11 XHFL/
10	YUFL/	P15V	10 XUFL/
9	SPAR	M15V	9 SPAR
8	GRND	GRND	8 GRND
7	GRND	GRND	7 GRND
6	PL5V	PL5V	6 PL5V
5	YHIL	SPAR	5 XHIL
4	YPLL	IPRT	4 XPLL
3	CUR1	FERT	3 CUR0
2	AVSS	FEOP/	2 AVSS
1	ANA3	IPOP/	1 ANA2

J2

	A	B	C
32	STRB/	MAI7/	32 GRND
31	DAI0	MAI6/	31 SEL0
30	DAI1	MAI5/	30 SEL1
29	DAI2	MAI4/	29 SEL2
28	DAI3	MAI3/	28 SEL3
27	DAI4	MAI2/	27 SEL4
26	DAI5	MAI1/	26 SEL5
25	DAI6	MAD5/	25 SEL6
24	DAI7	MAD4/	24 SEL7
23	ACKN/	MAD3/	23 GRND
22	BUFU	MAD2/	22 GRND
21	IPOS/	MAD1/	21 GRND
20	SP5V	PL5V	20 INIT/
19	GRND	EY	19 ERON/
18	IPOS	CONT	18 GRND
17	GRND	RSTR	17 DRDY/
16	CHASS	RDWR	16 HSB/
15	BUFU/	EX	15 SYNC/
14	SPAR	DAB0	14 SPAR
13	CHAS	DAB1	13 SP5V
12	IXD/	DAB2	12 ENBY/
11	RXD/	DAB3	11 ENAY/
10	RETS	DAB4	10 ENBX/
9	CLTS	DAB5	9 ENAX/
8	DASR	DAB6	8 SPAR
7	GRND	DAB7	7 DATR
6	BUFU/	INYD/	6 SPAR
5	ERDR/	INXD/	5 IOPS
4	INIT/	ENBY	4 BIXD
3	GRND	ENAY	3 BRTS
2	GRND	ENBX	2 AVSS
1	GRND	ENAX	1 ANA2


J5
PL5V A B IPOS/
PL5V 1 2 GRND
RSTR 3 4 CONT
EX 5 6 RDWR
DAB1 7 8 DAB0
DAB3 9 10 DAB2
DAB5 11 12 DAB4
DAB7 13 14 DAB6
J6
PL5V A B IPOS/
PL5V 1 2 GRND
RSTR 3 4 CONT
EY 5 6 RDWR
DAB1 7 8 DAB0
DAB3 9 10 DAB2
DAB5 11 12 DAB4
DAB7 13 14 DAB6
J7
PL5V 1 2 GRND
YJMI/ 3 4 XJMI/
YJPL/ 5 6 XJPL/
PREJ/ 7 8 STRT/
STEP/ 9 10 STOP/
HOME/ 11 12 HOLD/
HMYD/ 13 14 HMXD/
INIT/ 15 16 ENAX
IPLD/ 17 18 BFLD/
ERLD/ 19 20 EPLD/
I1LD/ 21 22 ENBX
F1LD/ 23 24 F2LD/
PL5V 25 26 GRND
J8
MD7 20 19 MMCS
MD6 18 17 MALE
MD0 16 15 MNT/SV2
MD5 14 13 IORD/
MD1 12 11 BUFU/
MD4 10 9 ERDR/
MD2 8 7 GRND
MD3 6 5 GRND
IOWR/ 4 3 +15V
MRESET 2 1 -15V
J9
ENAX 1 2 PL5V
GRND 3 4 ENAX/
ENBX/ 5 6 GRND
PL5V 7 8 ENBX
PL5V 9 10 SPAR
J10
10 SPAR
9 SPAR
8 FERT
7 FEOP/
6 IPRT
5 IPOP/
4 M15V
3 P15V
2 PL5V
1 GRND
J11
ENAY 1 2 PL5V
GRND 3 4 ENAY/
ENBY/ 5 6 GRND
PL5V 7 8 ENBY
PL5V 9 10 SPAR
J12
MA17/ 1 2 GRND
MA16/ 3 4 GRND
MA15/ 5 6 GRND
MA14/ 7 8 GRND
MA13/ 9 10 GRND
MA12/ 11 12 GRND
MA11/ 13 14 GRND
MA05/ 15 16 GRND
MA04/ 17 18 GRND
MA03/ 19 20 GRND
MA02/ 21 22 GRND
MA01/ 23 24 GRND
PL5V 25 26 GRND

SMCC CONNECTORS LISTING

J1 (JMACH) Connector (96 Pin DIN Connector)

PIN #	SYMBOL	FUNCTION	DESCRIPTION	NOTES
Row "A"				
A1	ANA3	INPUT	ANALOG INPUT 3	{Could be used {as analog speed {command. Use 0 {to +5V for 0 to
A2	AVSS	INPUT	ANALOG RETURN	{full scale.
A3	CUR1	INPUT	CURRENT 1 FEEDBACK	Current limiting
A4	YPLL	INPUT	Y AXIS + LIMIT SW.	Low enables the move
A5	YMIL	INPUT	Y AXIS - LIMIT SW	Low enables the move
A6	PL5V	I or 0	+5VDC POWER SUPPLY	{Could be used as {input to card or output to encoders.
A7	GRND	I or 0	+5 VDC RETURN	Logic GND
A8	GRND	I or 0	+5 VDC RETURN	Logic GND
A9	SPAR	SPARE		
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 PHASE 1, 2, 3, or 4	Nonbuffered pulse
A20	BPO6/	OUTPUT	PWM OUT	
A21	BPO5/	OUTPUT	PWM OUT	
A22	SPAR	SPARE		
A23*	PHA3/	OUTPUT	PHASE 3 RETURN	{Analog differential
A24	PHA3	OUTPUT	PHASE 3 COMMAND	{Signal Out Y axis
A25	M15A	INPUT	-15V SUPPLY	From power amplifier
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 amplifier
A30	AGND	INPUT	+/- 15V COMMON	From power amplifier
A31	AENA	OUTPUT	AMPLIFIER ENABLE	Programmable polarity

A32	FALT/	INPUT	AMPLIFIER FAULT	Low to A30 is Fault

*NOTE: If PHA3/ is not needed for Servo Amplifier, do not tie to AMP-common (AGND).

J1 (JMACH) Connector (96 Pin DIN Connector)

PIN #	SYMBOL	FUNCTION	DESCRIPTION	NOTES
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
B19	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/	INPUT	GO HOME COMMAND	Low is GO HOME
B24	HOLD/	INPUT	HOLD MOTION	Low is Hold
B25	STEP/	INPUT	STEP THROUGH PROGRAM	Low is Step
B26	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:
a) In position for this card "Low is true".
b) In position for this card "High is true".

c) MO1/output "Low is true".

J1 (JMACH) Connector (96 Pin DIN Connector)

PIN #	SYMBOL	FUNCTION	DESCRIPTION	NOTES
Row "C"				
C1	ANA2	INPUT	ANALOG INPUT 2	{Could be used as {analog speed command {Use 0 to +5V for 0
C2	AVSS	INPUT	ANALOG RETURN	{to full scale.
C3	CURO	INPUT	CURRENT 0 FEEDBACK	Current Limiting.
C4	XPLL	INPUT	X AXIS + LIMIT SW.	Low enables the move
C5	XMIL	INPUT	X AXIS - limit sw.	Low enables the move
C6	PL5V	I or 0	+5 VDC POWER SUPPLY	Could be used as 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 PHASE 1, 2, 3, or 4	Nonbuffered pulse
C20	BPO2/	OUTPUT	PWM OUT	
C21	BPO1/	OUTPUT	PWM OUT	
C22	SPAR	SPARE		
C23*	PHA1/	OUTPUT	PHASE 1 RETURN	{Analog differential
C24	PHA1	OUTPUT	PHASE 1 RETURN	{signal out X axis
C25	M15A	INPUT	-15 V SUPPLY	From power amplifier
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 amplifier
C30	AGND	INPUT	+/- 15 V COMMON	From power amplifier
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	FUNCTION	DESCRIPTION	NOTES
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	RS232	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 (BUSSED)	High is "In position"
A19	GRND	COMMON	SMCC COMMON	
A20	SP5V	OUTPUT	+5 VDC SUPPLY	Deactivated by E26
A21	IPOS/	OUTPUT	IN POSITION (BUSSED)	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	FUNCTION	DESCRIPTION	NOTES
Row "B"				
B1	ENAX	INPUT	X HANDWH ENC.A CH.	5V TTL square pulse
B2	ENBX	INPUT	X HANDWH ENC.B CH.	5V TTL square pulse
B3	ENAY	INPUT	Y HANDWH ENC.A CH.	5V TTL square pulse
B4	ENBY	INPUT	Y HANDWH ENC.B CH.	5V TTL square 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	MAO1/	INPUT	MACHINE INPUT 1	5V TTL Low is a true
B27	MAI2/	INPUT	MACHINE INPUT 2	5V TTL Low is a true
B28	MAI3/	INPUT	MACHINE INPUT 3	5V TTL Low is a true
B29	MAI4/	INPUT	MACHINE INPUT 4	5V TTL Low is a true
B30	MAI5/	INPUT	MACHINE INPUT 5	5V TTL Low is a true
B31	MAI6/	INPUT	MACHINE INPUT 6	5V TTL Low is a true
B32	MAI7/	INPUT	MACHINE INPUT 7	5V TTL Low is a true

J2 JPAR-RS232 Connector (96 Pin DIN Connector)

PIN #	SYMBOL	FUNCTION	DESCRIPTION	NOTES
Row "C"				
C1	PL5V	OUTPUT	+5V DC	Power supply out
C2	SYNC/	OUTPUT	SYNCHRONIZER	Low True from A0 card
C3	BRTS	OUTPUT	BUFFERED RTS	For RS232 daisychain
C4	BTXD	OUTPUT	BUFFERED TXD	for RS232 daisychain
C5	IPOS	OUTPUT	IN POSITION	

			(BUSSED)	High is "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 HWL 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 card
C16	HSB/	INPUT	READ STROBE	Low is "Read Strobe"
C17	ORDY/	OUTPUT	OUTPUT READY	Low is "Read Strobe"
C18	GRND	COMMON	SMCC COMMON	Low is "Read Strobe:
C19	EROR/	OUTPUT	SIGNALS ERROR	Low is an "Error"
C20	INIT/	INPUT	RESETS SMCC	Low is a "Reset"
C21	GRND	COMMON	SMCC COMMON	
C22	GRND	COMMON	SMCC COMMON	
C23	GRND	COMMON	SMCC COMMON	
C24	SEL7	OUTPUT	DATA7 OUTPUT	Scanner output for reading thumbwh 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	FUNCTION	DESCRIPTION	NOTES
1	PL5V	OUTPUT	+5V POWER	
2	GRND	COMMON	SMCC COMMON	
3	RSTR	OUTPUT	READ STROBE	TTL
4	CONT	OUTPUT	CONTRAST ADJUST.VEE	0 to 10 VDC
5	EX	OUTPUT	X DISPLAY ENABLE	High is "Enable"
6	RDWR	BIDIRECT.	READ OR WRITE	E27 selects func.
7	DAB1	BIDIRECT.	DISPLAY DATA1	5V TTL
8	DAB0	BIDIRECT.	DISPLAY DATA0	5V TTL
9	DAB3	BIDIRECT.	DISPLAY DATA3	5V TTL
10	DAB2	BIDIRECT.	DISPLAY DATA2	5V TTL
11	DAB5	BIDIRECT.	DISPLAY DATA5	5V TTL
12	DAB4	BIDIRECT.	DISPLAY DATA4	5V TTL
13	DAB7	BIDIRECT.	DISPLAY DATA7	5V TTL
14	DAB6	BIDIRECT.	DISPLAY DATA6	5V TTL

J6 JDISPY Connector (14 Pin Header)

PIN #	SYMBOL	FUNCTION	DESCRIPTION	NOTES
1	PL5V	OUTPUT	+5V POWER	
2	GRND	COMMON	SMCC COMMON	
3	RSTR	OUTPUT	READ STROBE	TTL
4	CONT	OUTPUT	CONTRAST ADJUST	0 to 10 VDC
5	EY	OUTPUT	X DISPLAY ENABLE	High is "Enable"
6	RDWR	BIDIRECT.	READ OR WRITE	E27 selects func.
7	DAB1	BIDIRECT.	DISPLAY DATA1	5V TTL
8	DAB0	BIDIRECT.	DISPLAY DATA0	5V TTL
9	DAB3	BIDIRECT.	DISPLAY DATA3	5V TTL
10	DAB2	BIDIRECT.	DISPLAY DATA2	5V TTL
11	DAB5	BIDIRECT.	DISPLAY DATA5	5V TTL
12	DAB4	BIDIRECT.	DISPLAY DATA4	5V TTL
13	DAB7	BIDIRECT.	DISPLAY DATA7	5V TTL
14	DAB6	BIDIRECT.	DISPLAY DATA6	5V TTL

J7 JPAN Connector (26 Pin Header)

PIN #	SYMBOL	FUNCTION	DESCRIPTION	NOTES
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	FUNCTION	DESCRIPTION	NOTES
1	-15V	OUTPUT	-15V POWER	For extension card
2	M RESET	OUTPUT	RESET	For extension card
3	+15V	OUTPUT	+15V POWER	For extension card
4	IOWR/	OUTPUT	WRITE DATA	Low is Write
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	
12	MD1	BIDIRECT.	ADDRESS DATA1	
13	IORD/	OUTPUT	READ DATA	Low is Read
14	MD5	BIDIRECT.	ADDRESS DATA5	
15	MWT/SV2	INPUT/	GENERAL INPUT/	(Not used in SMCC)
		OUTPUT	SERVO OUTPUT 2	(Controlled by E4)
16	MD0	BIDIRECT.	ADDRESS DATA0	
17	MALE	OUTPUT	ADDRESS LATCH ENABLE	
18	MD6	BIDIRECT.	ADDRESS DATA6	
19	MMCS	OUTPUT	BOARD SELCET	
20	MD7	BIDIRECT.	ADDRESS DATA7	

J9 JXHW Connector (10 Pin Header)

PIN #	SYMBOL	FUNCTION	DESCRIPTION	NOTES
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.	5V TTL sq. pulse
6	GRND	COMMON	SMCC COMMON	
7	PL5V	OUTPUT	+5V POWER	
8	ENBX	INPUT	X HANDWHEEL B CH.	5V TTL sq. pulse
9	PL5V	OUTPUT	+5V POWER	
10	SPAR	SPARE		

J10 Power Connector (10 Pin Terminal Strip)

PIN #	SYMBOL	FUNCTION	DESCRIPTION	NOTES
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	{see NOTE 1	Opto-isolated
7	FEOP	OUTPUT	FOLLOWING ERROR	Opto-isolated
8	FERT	OUTPUT	FOLL. ERROR RETURN	Opto-isolated
9	SPAR	SPARE		
10	SPAR	SPARE		

J11 JYHW Connector (10 Pin Header)

PIN #	SYMBOL	FUNCTION	DESCRIPTION	NOTES
1	ENAY	INPUT	Y HANDWHEEL A CH.	5V TTL sq. pulse
2	PL5V	OUTPUT	+5V POWER	
3	GRND	COMMON	SMCC COMMON	
4	HWAY/	INPUT	Y HANDWHEEL A CH.	5V TTL sq. pulse
5	HWBY/	INPUT	Y HANDWHEEL B CH.	5V TTL sq. pulse
6	GRND	COMMON	SMCC COMMON	
7	PL5V	OUTPUT	+5V POWER	
8	ENBY	INPUT	Y HANDWHEEL B CH.	5V TTL sq. pulse
9	PL5V	OUTPUT	+5V POWER	
10	SPAR	SPARE		

J12 JOPT Connector (26 Pin Header)

PIN #	SYMBOL	FUNCTION	DESCRIPTION	NOTES
1	MAI7/	INPUT	MACHINE INPUT 7	5V TTL, HIGH is True
2	GRND	COMMON	SMCC COMMON	
3	MAI6/	INPUT	MACHINE INPUT 6	5V TTL, HIGH is True
4	GRND	COMMON	SMCC COMMON	
5	MAI5/	INPUT	MACHINE INPUT 5	5V TTL, HIGH is True
6	GRND	COMMON	SMCC COMMON	
7	MAI4/	INPUT	MACHINE INPUT 4	5V TTL, LOW is True
8	GRND	COMMON	SMCC COMMON	
9	MAI3/	INPUT	MACHINE INPUT 3	5V TTL, LOW is True
10	GRND	COMMON	SMCC COMMON	
11	MAI2/	INPUT	MACHINE INPUT 2	5V TTL, LOW is True
12	GRND	COMMON	SMCC COMMON	
13	MAI1/	INPUT	MACHINE INPUT 1	5V TTL, LOW is True
14	GRND	COMMON	SMCC COMMON	

15	MAO5/	OUTPUT	MACHINE OUTPUT 5	Open Collector, True
16	GRND	COMMON	SMCC COMMON	LOW true
17	MAO4/	OUTPUT	MACHINE OUTPUT 4	Open Collector, True
18	GRND	COMMON	SMCC COMMON	LOW true
19	MAO3/	OUTPUT	MACHINE OUTPUT 3	Open Collector, True
20	GRND	COMMON	SMCC COMMON	LOW true
21	MAO2/	OUTPUT	MACHINE OUTPUT 2	Open Collector, True
22	GRND	COMMON	SMCC COMMON	LOW true
23	MAO1/	OUTPUT	MACHINE OUTPUT 1	Open Collector, True
24	GRND	COMMON	SMCC COMMON	LOW true
25	PL5V	OUTPUT	+5V POWER	
26	GRND	COMMON	SMCC COMMON	

Appendix G: Options And Accessories

G1: OPTIONS

1 Parallel data communications and thumbwheel (INC. AS STD) input capability.
2 Additional permanent memory (INC. AS STD) (8Kx8 EAROM)
3 Analog input conditioning
4 Additional RAM (32K RAM Buffer)

G2: ACCESSORIES

X1A Power supply (up to 3 SMCC's)
X1B Power supply (up to 5 SMCC's)

X2 +/- 15 VDC power supply for option X14

X3A 10 ft. RS232 cable
X3B 10 ft. parallel data cable
X3C 10 ft. RS232 and parallel cable

X4 Additional manual

X5 Mating connectors J7, J8, J9, J11, J12

X6 Handwheel encoder (with 6" leads)

X8A Input terminal board (insulation displacement type)
X8B Input terminal board (compression type)

X9 IBM executive software (MONITOR Program)

X10 8 bit parallel I/O board (for IBM PC)

X12A Position display (40x2 LCD)
X12B Position display 8 digit LED

X14 Additional card; 46 I/O, A/D convertor

X17 INPUT TERMINAL BOARD (has screw terminals for all input connections)

Contact Marketing Representatives for more detailed information

Appendix H: 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

H.1 Direct Commands

H.1.1 Card Address Select Command

An	n = 0 - 9, or a - f for card # 0 - 15. AA for all cards active simultaneously.

H.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.

H.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 then execute the next move.
$	Soft Reset.
$$	System Reset.
%nn,mm	Feed Rate % Override.
bnn	Goto Beginning of program if nn is omitted or go to subroutine nn.
PC=v	Set program counter,execution will start at line V.
l	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.

H.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.

H.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.

H.1.5 Advance 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.

H.1.6 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.

H.2 BUFFER INSTRUCTIONS

H.2.1 Program Flow Control Commands)

STOP	Sop 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/Setup 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.
FND	Jump out of the middle of a FOR/NEXT Loop.
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 current program line. nn is +/- 63.
GOTO nn Condition	GOTO specified subroutine, do not return.
GOSUB nn Condition	Jump to specified routine and return.

H.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 at which the "POT" command will change 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.
POT 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.

H.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.

H.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 i-PARAMETERS

Notation:

EC Encoder Counts
FR Ratio or Fractional Unit
[ ] Equivalent commands for Y Axis

i#	FUNCTION	RANGE	UNIT	DEFAULT
i00	Set Following Error Limit (FAULT)	0,+/-32767	EC	2000
i01	Set In-Position Band	0-65535	EC	100
i02	Set MDI, Manual Data Input Mode	0,1	None	0
i03	Set Hand Shake Enable	0,1	None	1
i04	Set Home Search Feed Rate	0 - 255	FR	0
i05	Set Servo Time	0 - 65535	mSec.	487
i06	Set Reference Max. Feed Rate	0 - 8388607	EC	2000
i07	Set Time Scale	0 - 65535	FR	65498
i08	Set Accel./Decel. Time	0 - 65535	1/1024 Sec.	512
i09	Set Position Integration Mode	0,1	None	1
i10	Set Following Error Limit	0 - 65535	EC	1000
i11	Set "Feed Hold" Slew Control	0,1	None	0
i12	Move Delay or Calculation Time	0 - 255	mSec	40
i13	MUST BE SET AS 0 FOR DC Motors	0	None	0
i14	Select Processor A/D Converter Mode	0 - 3	None	0
i15	Set Y Axis Disable in 2 Axis Mode	0,1	None	1
i16	Select Handwheel Operating Mode	0,1,2	None	0
i17	Set Velocity Display Time Base	0 - 2*31	mSec	15000
i18	Set Display Mode	0 - 3	None	3
i19	Set Highest Card Address	0 - f	None	0
i20[i40]	Set Proportional Gain Constant	0 - 65535	FR	50
i21[i41]	Set Differential Gain Constant	0 - 255	FR	240
i22[i42]	Set Velocity Feedforward Gain	0 - 255	FR	10
i23[i43]	Set Integral Gain Constant	0 - 255	FR	0
i24[i44]	Select Home Direction	0,1	None	0
i25[i45]	Set Home Offset	0,+/-32767	EC	500
i26[i46]	Set Home & User Flag Controls	0 - 7	None	0
i27[i47]	Set Hand Wheel Scale Factor	0 - 65535	FR	4096
i28[i48]	Select Hand Wheel Mode Control	0 - 3	None	3
i29[i49]	Hand Wheel Encoder Control Bit	0 -19[0-11]	None	3
i30[i50]	Set Acceleration Feedforward	0 - 255	FR	0
i31[i51]	Set Jog Feed Rate	0 - 255	FR	200
i32[i53]	Set Normal PWM Limit	0 - 255	FR	128
i33[i53]	Set Protective PWM Limit (I^2t)	0 - 255	FR	64
i34[i54]	Set Anti-Backlash Overtravel	+/-32768	EC	0
i35[i55]	Set Positive Position Limit	-1 to 2*27	EC	-1
i36[i56]	Set Negative Position Limit	-1 to 2*27	EC	-1
i37[i57]	Set 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]	Set Dead Band	0 - 255	FR	0
i62	PLC enable	0,1	None	0
i63	Set 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

INDEX

		Page
1/T Encoder period		8.7, A.13
A0		3.2
AA		3.5
Acceleration Feedforward Gain		A.14, 3.10
Acceleration/Deceleration time		A.4
Accessories - SMCC		G.1
Address Select Command		3.5
Altered Destination Move		5.12
Analog Command Input, Processor A/D Converter		3.12, A.6
Analog Input Output		2.2
Backlash compensation		A.15, 3.11
Backlash Take Up		A.18
Buffer Instructions		5.1
Card address DIP switch		3.2, C.1
Card address Select command		3.5
Circle4 mode		8.2
Circle Error Tolerance		A.18
cm, Count per Revolution		3.6
Control Character Commands		4.9
Controlled Hold mode (Feed Hold Slew Control)		4.18, A.5
Command		
$	Soft Reset	4.2
$$	System Reset	4.3
%nn,mm	Feed Rate Override	4.2, A12
=	Return to Pre-Jog	4.1
?	Send status and error codes	4.7
An	Address Select	3.5
AND	AND Condition/Action	6.1
AXnn[AYnn]	Absolute Move	5.10
bnn	Go to Begining	4.3
Bnn	Send Map	4.8
Bx	Send Map	4.8
CASE	Create jump table	5.5
CIRCLE N	Define circular motion	5.7
Cmm	Send address denifition	4.9
Cmm = A	Defines address	4.4
CNTRL	Control Character Command V M H	4.10
Cx	Send all address definition	4.9
dA,v	Read data	4.5
DELETE n, m	Delete program steps	4.5
DIVIDE n	Divide motion blocks	5.9
DLY n	Execute a delay time	5.13
DWELL nn	Wait	5.10
ELSE	Do if condition not true	6.1
END	End of PLC program	6.1

INDEX

		Page
END	End of program	4.1
f	Send following error	4.7
Fnn	Set Feed Rate	5.6
FOR n	Start loop	5.2
GOSUB nn	Subroutine call	5.3
GOTO (+/-) nn	Relative branch	5.3
GOTO nn	Subroutine branch	5.3
H	Feed Hold	4.1
h	Home	4.1
HOME	Go to home position	5.9
i00-i81	i-parameters	Appendix A
IF	Start of Condition	5.3, 6.1
innD	Send Default value	4.7
innH	Send Maximum allowable value	4.7
innL	Send Minimum allowable value	4.7
Innv	Set initialization parameter	4.4, 4.7
INSERT n	Insert program line	4.5
ix	Send all inn	4.7
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	5.7
LIST n,m	List program lines	4.5
Lnn	Label	4.1
mAA,d	write data	5.6
MAP n	Reallocate	4.5
mmAA,d	write data	5.6
Mnn	Send Mnn definition	4.9
Mnn = Cmm.b	Defines bit	4.4
Mx	Send all Mnn Definition	4.9
nAA,d	Logical AND	5.6
Next	End of loop	5.3
O(x)nn[OYnn]	open loop output	4.2
OR	OR condition/Action	6.1
p	Send position	4.7
PLC	List PLC Program	6.1
Pnn	Send Parameter value	4.4, 4.8
Pnn=v	Define Variable	4.4
POT nn	Feed rate override	5.9
PURGE	Purge PLC program	6.1
Px	Send Parameter value	4.8
q	Abort	4.1
Q	quit	4.2
q...	Altered Destination Move	5.12

INDEX

		Page
R	Run	4.1
R=nn	Set X and Y radii	5.7
R(X)[RY]=nn	Set Radius	5.7
rA,v	Read memory data	4.4
REPLACE n	Overwrite	4.5
RESET nn	Turn off M function	5.4
RESETP	Reset all P to zero	5.2
RETURN	Return from subroutine	5.1
s	Save program	4.4
S	Step	4.1
SET Cnn=aa	Define memory address	5.4
SET Mnn=Cmm.b	Define address location	5.5
SET nn	Turn on M function	5.4
SET Pnn	Set parameter value	5.1
SLEW nn	Controls slew rate	5.9
START	Start continous motion	5.1
STOP	Stop the program	5.1
THEN	Do this if condition true	6.1
TORQUE	Torque limited mode	5.7
tnn	Set Accel/Decel time	5.7
Tnn	Time mode	5.6
tt n	Triggered Move Calculation Time	5.12
uAA,d	Logical OR	5.6
Unn[Vnn]	Relative move	5.10
v	Send Velocity	4.7
W(X)[WY]	wait	5.10
wA,d	write data	4.5
WUT	Wait Until Trigger	5.10
X=nn[Y=nn]	Position set	5.6
xAA,d	Logical XOR	5.6
Xnn[Ynn]	Absolute move	5.10
y	Data Gathering	8.7
Z	Zero set	4.3
znn	Purge buffer	4.5, 4.4
^d	Go Until Interrupt	5.12
:d	Define ending velocity	5.10
Command format		3.4
Communication - parallel		3.2
Communication DIP switch		3.2, C.1
Connectors and PIN Assignments		Appendix F
Count per Revolution		3.6
CTS, RTS		3.2, E.3
Data Gathering		8.7
DB-25 connector Pin		3.3
Deadband		A.17, 3.11

INDEX

	Page
Diagnostics - Power up, communication	3.2
Differential Gain Constant	A.10, 3.10
Digital Outputs	2.3
Dip switch	C.1
Direct Commands	4.1
Display type	A.8
DMA communication	8.9
DMA, Host to Parabolic SMCC	8.9
DMA, Binary Parabolic SMCC, to Host	8.10
Mode control	A.17
DSR, DTR	3.3
Dulpex, HALF, FULL	3.3
E2, E3	3.1, 3.2, E.1
E7, E8	3.1, E.1
E Points, E1 - E50	E.1
External Pulse (Handwheel) Input	3.11, A.7
External RAM address definition	D.10
Feed Hold Slew control	A.5
Feed Pot mode	A.6
Following Error Limit	A.1, A.4
Go Until Interrupt	5.12
Hands-on Command example	4.10
Acceleration/Deceleration	4.14
Card Selection and Basic Move	4.10
Dual Axis Control	4.18, 5.17
Dual Axis Coordinated Motion	5.16
Following error	4.15
Jog Speed	4.12
Machine Input Output Functions	5.19, 5.20
Resetting SMCC	4.16
Subroutine	5.15
Time Specified Move	5.14
Unit of Position	4.11, 4.17
Usage of Buffer Instructions	4.15, 5.13
Elliptic Motion	5.18
Multiple SMCC Coordinated Motion	5.18
Handshake Enable	3.8
Handshake Enable in Host Communication	A.2
Handwheel Encoder Control bits	A.13
Handwheel Mode Control	A.13
Handwheel Operating mode	A.7
Handwheel Scale factor	A.11
Highest Card address	A.9
Home direction	A.10
Home Feedrate	A.2

INDEX

	Page
Home Flag and User flag control bits	A.11
Home Offset	A.11
Host communication - set up	3.1
In-position band	A.1
Instruction Format	3.4
Integral Gain Constant	A.10
Internal RAM address definition	D.3
J1 - J12 Connectors	F.1
JMACH, J1 connector	F.2
Jog feedrate	A.14
Jog Speed Control Mode	A.6
JPAR-RS232, J2	F.5
LED Display codes	B.1
M-Function I/O	4.9
Machine Input address	D.1, D.2
Machine Input, Output	4.9
Machine Output address	D.1, D.2
Manual Control Inputs	2.3, D.1
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, A.5
Motor/Amplifier connection	3.1
Move calculation time	A.5
Normal Data Processing Mode	3.8
Null Modem	3.1
On-board Dip switch	C.1
Operations	3.14
Options - SMCC	G.1
P variable arithmetic	5.2
Panel Interface Address	D.1, 2.3
Parabolic Profiling	8.1
CIRCLER4	8.1
direct	8.1
point-to-point S-Curve	8.4
S-Curve	8.2
PID controller	A.9
PLC action format	6.3
PLC condition format	6.2
PLC direct commands, buffer instructions	6.1
PLC Enable	A.17, 3.14
Point to Point S curve Move	8.4
Position and Display Scale and Format	A.16
Position Encoder Mode control bits	A.16

INDEX

	Page
Position Integration	3.10, A.4
Position Limit, Range	A.15
Position Integration Mode	A.4
Position Offset Mode	A.13
Processor A/D Converter mode	A.6
Program Control Command	4.2
Program Definition/Control Instructions	5.1
Program Editing Comands	4.5
Proportional Gain	3.10, A.9
PWM Limit control	A.14
Range Mode Control	A.17
Rapid Hold Mode	4.18, A.5
Reference (Maximum) Feedrate	A.3
Reset SMCC	4.17
RS-232 Port	2.1
RTS, CTS	3.3, E.2
RXD, TXD	3.3, E.2
S curve	8.2
Servo perofrmance	2.1
Servo Time period	A.3
Set up Encoder	3.6
Set up Home	3.13
Set up Initialization command	3.5
Set up Limit	3.7
Set up Position/Velocity Display	3.12, A.16
Set up System Reference, i06, i08, i31[i51]	3.6
Shortest Distance	A.17
SMCC	1.1
Software positive negative limit	A.15
Specification - SMCC	2.1
Status Byte	B.1, 4.7
Status codes	B.1
Status Query Commands	4.7
Summary of i-parameters	H.5
Summary of SMCC Commands	H.1
SYNC clock	3.1
Test Points, TP1 - TP11	E.7
Three point smoother	8.2
Time move	5.6
Time scale	A.3
Timer usage (PLC)	6.4
Tuning	3.8
TXD, RXD	3.3, E.2
User flag	A.11
Velocity control	2.2

INDEX

	Page
Velocity control mode	A.6
Velocity display time base	A.7
Velocity feedforward	A.10
Version number address	D.1
Wait Until Trigger	5.10
Y axis disable	A.6