DELTA TAU DATA SYSTEMS

```
****************************
*  *********************  *
*  *                   *  *
*  *       SMCC        *  *
*  *                   *  *
*  *********************  *
****************************
```

IBM P.C. BASED PROGRAMMING SOFTWARE WRITTEN IN "C"

For Use With The Smart Motor Controller Card
(SMCC, P/N 602090-100)

This manual may be used with
SMCC Version 3.6

February 7, 1989

Delta Tau Data Systems, 21119 Osborne Street, Canoga Park, CA 91304
PH: (818)998-2095 FX: (818)998-7807

TABLE OF CONTENTS

.pa

1.0 SMCC VERSION 3.6

SMCC V3.6 is an all-purpose programming tool for the Smart Motor Controller Card (SMCC). It allows for on-line terminal operations, program editing, and data gathering and graphing capabilities. It is written in the "C" programming language.

SYSTEM REQUIREMENTS

SMCC Version 3.6 runs on an IBM PC, PC/XT, PC/AT, and SYS 2 (AT and SYS 2 users please see note at end of manual), or compatibles with at least 512K and one disk drive. In addition, one standard serial port or one bi-directional parallel port is required if the SMCC card is external to the PC. (A half-size card which plugs directly into the PC's bus and specifically provides bi-directional parallel communication may be obtained directly from the factory: ask for SMCC Accessory 10.) The new SMCC-PC card fits into the backplane of the PC, communicating directly on the computer's own bus, and so does not need an interface port.

COPYING THE SUPPLIED DISKETTE

Making a backup copy of your diskette is essential for insuring that the SMCC programs are not destroyed if the original diskette is ever erased or damaged.

If you have two floppy disk drives, you may backup the original SMCC diskette as follows:

1. Insert your DOS diskette into Drive A and a blank diskette into Drive B and reset the computer.

2. At the A> prompt type FORMAT B: and press <ENTER>.

3. After the disk in Drive B is formatted, type:
   DISKCOPY A: B: and press <ENTER>.

4. Remove the DOS diskette. Place the SMCC diskette in Drive A and press <ENTER> to the prompt.

5. After the original disk is copied, store it in a safe place and use the backup copy as your working master.

If you have a hard disk drive, you may back up the original SMCC diskette as follows:

1. At the C> prompt type: MD SMCC and press <ENTER>.

2. Type: `CD SMCC` and press <ENTER>.
3. Type: `COPY A:*.*` and press <ENTER>.

Note: If you do not want to copy the system files on the supplied diskette, you may copy the other files individually onto the hard disk: SMCC.EXE, the desired graphics driver (xxx.BGI -- see below), and any sample files. The supplied diskette operates under DOS Version 3.2.

From now on, to run SMCC from your hard disk, type `CD \ SMCC` and press <ENTER>. Then type: `SMCC` and press <ENTER>.

The supplied diskette provides a number of 'graphics driver' files on the disk. These files, which have the extension '.BGI' (binary graphics interface) appended to their names, allow the program to utilize various graphics displays. For a given computer, you will only need one of these driver files, and you may not want to copy the rest onto your working disk, whether it is hard or floppy. The names of the driver files are suggestive of the display they control:

```
CGA.BGI -- Color Graphics Adapter
EGAVGA.BGI -- Enhanced Graphics Adapter &
               Video Graphics Adapter
HERC.BGI -- Hercules Graphics Adapter
ATT.BGI -- AT&T 400-line Graphics Adapter
PC3270.BGI -- IBM 3270 PC Graphics Adapter
IBM8514.BGI -- IBM 8514 Graphics Adapter
```

RUNNING SMCC

The SMCC terminal program is provided with the basic MS-DOS V3.2 system on the diskette. This allows you to boot the computer with the SMCC diskette as the system disk. To do this, place the SMCC diskette into Drive A of your computer and turn the computer on. If your computer is already on, you may reset it by holding down the <Ctrl> and <Alt> keys and pressing <Del>. (If you are using a hard disk drive, follow the instructions in the section above.)

When the disk drive light turns on, you will be prompted for the date and time. Afterward, SMCC Version 3.6 will load automatically.

If you are already running and do not wish to reboot the computer, simply specify the disk with the SMCC Terminal Program as the current disk (and subdirectory, if necessary) and type 'SMCC' at the prompt. The program will load.

After the two windows appear on the screen and a listing of the function keys appears in the right-hand window, you will be prompted with a question such as the following:

```
System Configured as COM1 at 9600 baud.
With X1 Clock Speed
```

Do you want to change it? (y/n)? __

If your Smart Motor Controller Card is configured for this setting, simply press <N> or <ENTER>. If you press <Y>, you will be allowed to configure SMCC to the proper port type (serial, parallel, or PC bus), baudrate (9600, 4800, 2400, 1200, 300, 38.4k, 19.2k), or port address, etc. which you are using. Please refer to the SMCC manual for the dip switch settings which configure your card for the proper address and serial or parallel communication. Also, refer to the Alt-F3 (Configure) key description in Section 2 of this manual for detailed instructions, and the 'Note for IBM AT Users in Section 7 for explanation of clock speed selection.

SMCC proceeds to poll the available cards (A0 through Af) and determines which cards, if any, are connected to your system. You will then be placed into Terminal Mode. If no cards are connected, or the communication port in your computer is not properly configured, you will get a message: '0 Cards found'.

Pressing the "Escape Key" allows you to exit SMCC and return to DOS. If you wish to try to specify the interface configuration again, use the ALT <F3> Key. SMCC will enter its normal terminal mode regardless of whether any SMCC cards are connected to the computer.

.pa

2.0 SMCC TERMINAL MODE

After SMCC successfully determines that one or more cards are connected, you may proceed to communicate with the SMCC cards. The left-hand window will act as a direct terminal to the card. Anything you type will be sent to the card (once you press <ENTER>, and anything sent from the card will be promptly displayed on the screen.

Example: Typing the letter "J" and pressing <ENTER> will cause the motor attached to the card to jog in the positive direction. Type another "J" or "Q" to stop the motion. Typing the lowercase letter "p" will cause the Smart Motion Control Card to send its position data which will be isplayed on the next line of the screen.

Note that upon entering terminal mode for the first time, SMCC automatically addresses card A0. Thus the position data displayed by the "p" command will be the position of card A0. To change the card being addressed, type "A" and the card's number (0-9) or letter (a-f) and press <ENTER>.

In addition to the on-line commands described in the Smart Motor Controller Card Manual, the following function keys apply:

<F1> - I-PARAMETERS

This function key retrieves up to 99 (ninety-nine) i-parameters of the card currently addressed. The parameters are divided into five screens. To display the next screen, press the <Pg Dn> key located on the numeric keypad. To display the previous screen, press the <Pg Up> key located on the numeric keypad. Once the parameter you wish to modify is displayed on the current screen, you may access it by positioning the highlighted bar over the parameter. To move the highlighted bar, use the Up Arrow and Down Arrow keys, and to modify the parameter, press <ENTER>.

After positioning the bar over the desired parameter and pressing <ENTER>, the following message will appear in the right-hand window:

```
CHANGE - ixx

Maximum:  xxxxxx
Minimum:  xxxxxx
Default:  xxxxxx
Current:  xxxxxx

ENTER NEW VALUE:
->
```

The values listed refer to the appropriate values of the parameter you selected. You may enter a new value, or press <ENTER> to cancel the change. If you enter a value which exceeds the critical values of the parameter, a warning message will flash and you will be prompted to try again.

When you finish reviewing the i-parameters, press <Esc> to return to Terminal Mode, or "P" to print the i parameters, or "S" to save the parameters.

<F2> - P-VARIABLES

This key clears the terminal window and displays the fifteen P-variables. For more information on P-variables, consult the Smart Motor Controller Card Manual.

<F3> - C-DEFINITIONS

This function key displays the sixty-four C functions in four symmetric columns. For more information on C functions, consult the Smart Motor Controller Manual.

<F4> - M-DEFINITIONS

This function key displays the sixty-four M functions in four symmetric columns. For more information on M functions, consult the Smart Motor Controller Manual.

<F5> - LIST PROGRAM

This key will display the buffer of the card currently addressed. You are prompted for the starting line number, after which each carriage return keystroke will display a page full of consecutive program lines. To return back to Terminal Mode, press <ESC>.

<F6> - ARROW JOG

This function key allows you to jog the X (and Y) axis of the current card in both the positive and negative directions. Upon pressing the key, a map of the arrow keys is displayed. Pressing the Up Arrow key will jog the Y axis in the positive direction. Similarly, Down Arrow will jog the Y axis in the negative direction, Left Arrow jogs the X axis in the negative direction, and Right Arrow jogs the X axis in the positive direction. To stop the jog, press the key which activated the jog or any other key on the keyboard. Pressing <ESC> will stop any jog and return you to Terminal Mode. Pressing the <END> key on the keyboard will also stop jogging and pressing the <HOME> key on the keypad will issue an "=" command to the card, which will cause it to move back to its prejog position.

<F7> - CARD STATUS

This key reads the thirty-two status bits (standard SMCC), or the sixty-four status bits (parabolic SMCC) of the card currently addressed, and displays the current state of each bit with its label. For more information on the meaning of the status bits, consult the Smart Motor Controller Card Manual.

<F8> - DISPLAY POSITION

This function key displays the synchronized position for all cards connected to your system regardless of which card is currently addressed. The "Control V" command is used by SMCC to freeze all card positions simultaneously, which are subsequently read and displayed.

<F9> - SUBROUTINES

This key displays a map of all the possible subroutines and marks the locations (line number in the buffer) of the ones currently active. The buffer size is also displayed at the top of the screen.

<F10> - FILE EDITOR

This key allows you to create, edit, send, and retrieve SMCC programs. See section 3.0 below for detailed instructions.

.PA
The following functions are activated by holding down the <Alt> key and pressing the appropriate function key. The alternate function key selections can be viewed on the right-hand screen by holding down the <Alt> key and pressing <Q>.

Alt <F1> - RUN MODE

This function key allows you to continuously send a RUN command to the connected cards. After you specify a delay time, SMCC sends a RUN command to all the available cards. When SMCC determines that all the cards are in position again, it waits the specified delay time and then sends another RUN. This process is repeated until the <Esc> key is pressed. Note that the delay time has an accuracy of 1/100th of a second and must be specified in the decimal format. Ex. 0.05, 1.17, 5.00.

Alt <F2> - O COMMAND

This function key allows you to send a series of 'O' commands (open loop outputs) to the currently addressed card. This can be very useful for adjusting or checking the tuning of an amplifier. See Section 4.0, below.

Alt <F3> - CONFIGURE

This function key allows you to reconfigure and re-poll your system. You may change the communication parameters (port type, baudrate, etc.) and repoll the available cards. Note that this function is automatically invoked when SMCC is first run.

The current interface configuration is shown, and you are asked if you wish to change it. If you answer 'Y' or 'y', you will be asked which of the following interfaces you wish to configure:

```
     <1>  COM1   <2>  COM2   <3>  PARALLEL   <4>  PC BUS
```

where selections 1 and 2 refer to the standard RS-232 ports on the PC, selection 3 refers to Delta Tau's bidirectional parallel interface card to an external SMCC board, and selection 4 refers to a direct interface on the computer bus to an SMCC-PC board installed in the backplane.

If you selected the RS-232 interface, you will be asked to select the baud rate from a menu of choices. The rate you choose must match the setting on the DIP switch on the SMCC card(s), and, in general, the longer the cable is, the lower the baud rate should be (see Section 7, below, and the SMCC board manual for details).

If you selected the parallel interface, you will be asked to select the address of the interface card from a menu of choices. The address you choose must match that set by the DIP switch on the interface card (see the SMCC manual for details).

If you chose the PC Bus interface for the SMCC-PC card, you will be asked to specify the address of any card(s) installed. To talk to this program, the cards must be numbered consecutively starting with A0. For example, if you have four cards, they must be numbered A0, A1, A2, and A3 (set by DIP switches on the cards). The bus addresses of the cards are set by another DIP switch on board. Starting with A0, you must enter the address of each card until you have specified all of the cards. When asked for the address of the next card past your highest numbered card, simply type <ENTER> or <ESC> at the prompt, and the sequence will stop.

Regardless of which interface you chose, you will be asked:

```
     Enter clock speed multiplier (1-9): __
```

This puts a limit on the time the program waits for a handshake bit for reading or writing. The faster the clock rate for the computer, the higher the number you want to enter here, so as not to outrun the interface speed. See the notes in Section 7, below, for details.

Alt <F4> - TUNE FILTER

This function key puts you in a section of the program that allows you to adjust the parameters of the SMCC's digital filter for the card currently addressed. See section 5.0, below, for detailed instructions and examples.

Alt <F5> - LIST CARDS

This key will display the card currently addressed as well as all the available cards connected to your system.

Alt <F6>- DISK ON

This key will toggle the DISK ON option. With DISK ON, all data received from the card will be echoed to the default disk drive under the filename "TERMDATA.SMC." This file may be examined by loading it through the SMCC editor. When SMCC is first run, if the file TERMDATA.SMC already exists, it will be erased. However, all subsequent uses of DISK ON will append all incoming data to the file.

Alt <F7>- PRINTER ON

This key will toggle the PRINTER ON option. With PRINTER ON, all data received from the card will be echoed to any standard line printer as well as to the screen. Make sure that the line printer has paper and is On-Line.

Alt <F8>- DISPLAY OFF

This key stops data from being displayed on the screen.

Alt <F9>- SYS COMMAND

This key permits accessing all IBM P.C. DOS Utility functions. Pressing <ESC> or <ENTER> in response to a prompt returns you to the SMCC Terminal Program just as you left it.

Alt <F10>- GATHER/PLOT

This key puts you in a program section that allows you to gather real-time servo data from the card, and to plot, print, display, and store this data. See section 6.0, below, for detailed instructions.

.PA

3.0 SMCC EDITOR

The editor is a word processor for SMCC programs of up to 65000 characters. In this section of the program, lines typed in the left window are not sent to the SMCC card(s), but instead entered into the editor buffer. The line number corresponding to the line of the editor buffer to which the cursor is currently pointing is displayed in the right window. Movement of the cursor is possible with the four Arrow Keys. The following editing keys also apply:

PgUp -	Displays the previous page (23 lines) of the current program.
PgDn -	Displays the next page (23 lines) of the current program.
Home -	Displays the first page (23 lines) of the current program.
End -	Displays the last page (23 lines) of the current program.
Alt-I -	Inserts a blank line at the current cursor position. All subsequent lines are moved down.
Alt-D -	Deletes the line at the current cursor position. All subsequent lines are moved up.
Alt-C -	Clears the current buffer and filename.
Ins -	Toggles between the "INSERT ON" and "INSERT OFF" modes of the editor. Insert Off allows you to type over previously entered text, while Insert Off will move any previously entered text to the right of the cursor.

Note that each line is limited to fifty-two characters and the editor is limited to 65000 characters. The editor will signal you when the last line is reached.

In addition, the following function keys may be used:

<F1> - FILE UTILITIES

This key allows you to load, save, insert, or delete a program on the current disk. In addition, you may print the file currently in the editor or a file saved on a diskette. You will be presented with the following selection menu in the left window:

```
     <1> - Load File          <4> - Delete File
     <2> - Save File          <5> - Print Current File
     <3> - Insert File        <6> - Print Disk File
```

Enter the number of your selection at the prompt. The selections perform the following tasks:

<1> - Load File : Allows you to bring a file in from disk to the editor buffer. Any program currently in the editor buffer will be erased.

<2> - Save File : Allows you to save the program currently in the editor buffer to disk.

<3> - Insert File : Allows you to bring a file in from disk and insert it into the program currently in the editor buffer at the present cursor location.

<4> - Delete File : Allows you to delete a disk file.

<5> - Print Current File : Allows you to print the file currently in the editor buffer to the standard printer for the system.

<6> - Print Disk File : Allows you to print a file on disk to the standard file for the system.

Note that filenames are limited to eight standard characters and unless you specify your own extension, the extension ".SMC" is automatically added to any filename. Disk and subdirectory specifiers can be entered in front of a filename, up to a total of 40 characters. The default filename for a newly created program in the editor buffer is NONAME.SMC.

<F2> - SYSTEM COMMAND

Provides access to the DOS operating system. Pushing <ESC> or <ENTER> in response to a system prompt returns you to where you left off.

<F3> - GET PARAMETERS

This key will upload the i-parameters from any card and append them to the program currently in the editor's buffer. You may specify the card address of the parameters to be loaded. Note that an extra line containing the card address will precede the parameter list in the editor's buffer.

<F4> - GET PLC BUFFER

This key will upload the PLC program from any card and append it to the program currently in the editor's buffer. You may specify the card address of the PLC program to be loaded. Note that an extra line containing the card address and a "PUR" will precede the PLC contents in the editor's buffer. This is done to ensure that when you later download the program to the SMCC Card, the PLC buffer will be cleared. For more information on the "PUR(GE)" command, consult the SMCC card manual.
.PA

<F5> - UPLOAD BUFFER

This function key uploads the program buffer from any card and appends it to the program currently in the editor's buffer. You may specify the card address of the buffer to be loaded. Note that an extra line containing the card address and a "z" will precede the card buffer contents in the editor's buffer. This is done so as to insure that when you later download the program to the SMCC Card, the buffer will be cleared. For more information of the "z" command, consult the Smart Motor Controller Card Manual.

<F6> - DOWNLOAD BUFFER

This key sends the entire contents of the editor's buffer to the card and returns you to Terminal Mode. Note that the editor's buffer is not erased; you may access it again by pressing <F10>.

.pa

4.0 'O' COMMAND (AMPLIFIER TUNING) DIRECTIONS

This section of the program allows you to program a repetitive sequence of open-loop outputs from the SMCC card using its 'O' command feature. This is very useful for tuning amplifiers, particularly those with tachometer feedback. It can take the place of special hardware devices that perform the same purpose. In addition, it has safety features that no hardware device can have: automatic shutdown on exceeding (programmable) velocity and position limits. It also uses data gathering and plotting to display the results of the stimulation (velocity, position, and acceleration versus time), eliminating in many cases the need for an oscilloscope to do the tuning.

The command sequence is very flexible; you can enter the output levels, their durations, the number of cycles, and the shutdown limits. You can repeat the last command sequence without entering the numbers again. You also have the capability to stop the sequence immediately by hitting any key on the computer keyboard.

SPECIFYING THE COMMAND SEQUENCE

After entering this section (by pressing ALT-F2 in the main Terminal program menu, you will see the heading 'O' COMMAND SPECIFICATION, and you will be asked if you wish to repeat the last sequence (press 0) or specify a new sequence (press 1). (If this is the first command sequence during a run of the program, the 'last' sequence has levels of +10 and -10, each held for 100 milliseconds, with 100 milliseconds of commanded zero between each level. There will be 20 cycles, with data storage on the first two cycles. The velocity limit is 1000, and there are no position limits. Explanations of these numbers follow below.)

If you choose to specify a new command sequence, you will be asked a series of questions to set the parameters for the sequence. With each question, the program will show you the valid range for that parameter, and the present value of the parameter. The response is always numerical. You may accept the present value simply by pressing <ENTER> in response to the question. If you wish to abort the whole process, press <ESC> in response to any question. You will be immediately returned to the main program level, with no command sequence executed.

If your SMCC card is configured for two axes (i13 = 0), the first question asked will be which axis you wish to command. Answer with '1' for X axis, or '2' for Y axis.

The next questions ask you to specify the O-command levels. The valid range for these levels is from -255 to +255. You have two levels to specify; usually you will want to set one positive and one negative to see how the amplifier works in both directions. (In addition, you can have zero levels to see the decaying behavior from both sides.)

Now you will be asked how long you wish to hold the output at each level. Your response should be in milliseconds. Following this, you will be asked how long you wish to have a zero command between each of the levels you have specified. The units are the same. If you specify zero here, there will be no zero command between levels. If you specify a non-zero value, the sequence for a cycle that you will get is: 1st level, zero, 2nd level, zero.

The next section asks you how many of these cycles you wish to perform. Entering zero here means the sequence will cycle indefinitely until it is stopped externally. Otherwise, it will do the number of cycles specified and stop without user intervention.

Following this, you will be asked at what level of velocity you want an automatic termination of the sequence. This is intended to protect the system by preventing it from ever moving too fast. This feature depends on having proper encoder feedback to the card, of course. The units for this limit are set by the i17 parameter on the SMCC, which also controls the velocity display units. The velocity is defined as the change in position over a "sample period", where i17 is one-half of the "sample period" in microseconds for the standard SMCC, or the "sample period" itself in microseconds for the parabolic SMCC. Entering zero for this parameter means that there is no velocity limit. The sequence is terminated if the velocity limit is exceeded in either the positive or negative direction.

Next you will be asked for the negative and positive positions at which you wish an automatic termination of the sequence. This also is a protective feature, and also depends on proper encoder feedback. The units for these limits are set by the i38 (X-axis) or i58 (Y-axis) parameter on the SMCC, which also controls the position display and data input format for that axis. The i38 / i58 parameter converts data as follows:

Format: n:m.d

$$\text{Limit \& Display units} = \text{Encoder counts} \times \frac{m}{n} \times 10^{-d}$$

(Descriptions of this parameter in versions of the SMCC manual dated 1988 and earlier are incorrect. This is the correct description.) The default value of 1:1.0 provides no conversion: you are working with the encoder counts themselves.

i38 / i58 Conversion example: Suppose you have a system with 2540 full encoder lines per inch. Using 4x decoding, this gives you 10160 encoder counts per inch. If you wish to work with your units in inches, with three places to the right of the decimal point to get a resolution of milli-inches, you would set a value of 10160:1000.3 for the parameter. Dividing the encoder counts by 10160 converts to inches; multiplying by 1000 brings it to milli-inches, and the '3' term puts the decimal point in the proper place.

For the command sequence limit, only integer values may be entered, and entering zero for a limit means no position limit in that direction (if you actually wish to stop if you go near zero, you should enter +1 or -1 for that limit). It is possible to enter a negative limit greater than the positive limit, but (provided neither limit is zero) the sequence will shut down immediately after it starts.

If you have a parabolic SMCC, you will then be asked if you wish to gather data while you are doing the command sequence. If you answer positively, the program will ask you for how many cycles you want to gather the data. You may collect data for 20 cycles or the total number of cycles you are performing, whichever is less.

RUNNING THE SEQUENCE

After you have finished answering these questions, you will be notified that the command program has been loaded and is ready to go. Prepare any tools that you need to analyze the response, then hit any key on the keyboard to start the sequence. The sequence will run until it is stopped by one of the following:

- specified number of cycles performed
- velocity limit exceeded
- either position limit exceeded

- any key on the keyboard pressed
- hardware shutdown (e.g. reset or power down)

You will be notified on the screen of the end of the sequence for any but the last of the reasons. If you have collected data, this data will be brought into the computer. You will then be able to plot the data just as under key <F4> of the data gathering and plotting section of the program.

At the end of this section of the program, you are returned to the main level of the program, operating as a terminal in the left window, with the function keys listed in the right window.

.pa

5.0 SMCC FILTER TUNING PROCEDURE

This section of the SMCC Terminal Program is intended to assist the user in setting the control parameters of the digital motion filter, a process often known as 'filter tuning'. The goal here is to achieve a stable and well-behaving system. The process is interactive and iterative: try a setting, see how it works, and adjust, until satisfied. The procedure was set up so that no special control expertise is required; the key measures used are easy to understand. The process requires the parabolic SMCC-P card because it needs the data logging capabilities of that card.

The SMCC-P filter tuning process is divided into two broad steps: first, adjusting the feedback parameters using step response information; and second, setting the feedforward parameters by observing the behavior of a profiled parabolic move. A filter tuned in this manner will be stable and quick-responding, and will track a desired move with negligible following error.

PREREQUISITES

In order to run this section of the program, you must already have the card set up and able to communicate properly with the host computer, the motor, amplifier, and encoder (or resolver). If you are using the SMCC-P's commutation to drive a brushless or induction motor, you must already have set up the commutation parameters successfully. The amplifier must already have been tuned, particularly if it utilizes a tachometer (refer to setup instructions in the SMCC manual and/or amplifier instructions). Once you have set up the motor commutation and tuned the amplifier, this procedure will work for all types of motors the SMCC can drive. You must bring the system to a location where significant travel is possible in each direction, particularly in the positive direction (moves start positive, then return to the starting position). If you lack confidence that you have the system set up correctly, you may wish to try this procedure first with the motor disconnected from the load. The values that you get will not be appropriate for the entire system, but you will get a good feeling for how the procedure operates without risking damage to your load.

ENTERING AND OPERATING THE TUNING PROCEDURE

To start the tuning procedure from the main SMCC Terminal Program, first make sure you are addressing the card you wish to tune, then hit function key ALT-F4. This is labeled 'Tune Filter' in the alternate main menu. After hitting this key, you will see a screen similar to the one you were just in, but headed 'SMCC Tuning' and with a different function key menu. In the main (left) window, you will see an introductory message, and a question as to whether your amplifier uses a tachometer. This question is only important if you wish to use the default starting filter, which differs depending on the type of amplifier. Answer this question with a 'y' or 'n'.

At this point, the program acts just like the main part of the program. If you type alphanumeric keys, the computer acts just like a terminal connected to the SMCC, sending and receiving command and data lines. The function keys are for more sophisticated tasks. As in the main program, there are two sets of function keys, regular and alternate. You can toggle between the menus for these sets by hitting Alt-Q. It does not matter which menu is showing -- you can use any of the function keys.

In the right window below the function key menu, you will see the current values of the feedback and feedforward parameters for the card and axis presently addressed. Note: In order for these displayed values to be correct at all times, you must change these parameters only through use of the function keys. Below this you will see the card and axis information. If your card is set up for more than one axis, you can change the addressed axis by hitting F8.

After ordering the card to do a move, that move will be plotted to the screen. When the procedure is in this graphics mode, it operates somewhat differently. The right window looks almost the same, and the function keys work just as they do in text mode. However, most alphanumeric keys have no effect here. The only ones that do anything are:

```
     'p' or 'P': Print the graphics screen to the printer
                 (your printer must be graphics capable)
     'r' or 'R': Return to the text screen
     SPACE     :  'Kills' the SMCC card.  This is to be used when
                 the board gets out of control, as with an un-
                 stable filter.  It sends a 'q' and a 'k' to the
                 card, and reports to you what it has done.
                 (SPACE does not do this in the text screen
                 except while it is preparing for a plot; other-
                 wise it only sends a space.  Type 'q'<ENTER>
                 'k'<ENTER> if you need to kill the system while
                 in text mode.)
```

The ESC key returns you to the text screen, just as 'R' and 'r'.

FEEDBACK TUNING WITH STEP RESPONSE

Step response is often used as a method of evaluating a feedback filter. Many controls textbooks contain information on interpreting step responses for establishing proper feedback, particularly for 'second-order' systems (current-controlled motors driving inertial loads -- the bulk of SMCC-P applications -- are second-order systems). In a step response, a sudden change is made to the command position, and the feedback filter attempts to bring the system to this new position. In observing how the system gets to the new position, we can deduce a great deal about the properties of the system. (It does not matter that you will not ever create such a large instantaneous step in position in the actual operation of your system -- the purpose of this 'jolt to the system' is to bring out system characteristics that might otherwise not be obvious.)

The SMCC-P has three feedback parameters to be adjusted in this process:

```
     Kp -- Proportional gain -- i20  [i40 for y-axis]
     Kd -- Derivative gain --   i21  [i41]
     Ki -- Integral gain --     i23  [i43]
```

The complete servo filter equation is:

```
     Command(n) = K0 * Kp * {EP(n) - Kd*AV(n) + [Ki/4096]*IP(n) +
                  Kvff*DV(n) + 4*Kaff*[DV(n)-DV(n-1)]}
     where:
             K0 = 1.25 volts/65536 (scale factor to real world)
          EP(n) = Position error for cycle n
          AV(n) = Actual velocity for cycle n
                  n-1
          IP(n) = SUM EP(j)  (integral of the position error)
                  j=0
           Kvff = Velocity feedforward gain (see below)
          DV(n) = Desired velocity for cycle n
           Kaff = Acceleration feedforward gain (see below)
        DV(n-1) = Desired velocity for cycle n-1
```

We will be looking at three key step response parameters to set the feedback:

```
     rise time -- the time it takes for the system to go from 10%
                  to 90% of the commanded step (natural frequency
                  is directly related to this).
     overshoot -- the percentage past the commanded step that the
                  system travels (damping ratio is directly re-
                  lated to this).
     settling time -- the time it takes for the system to get and
                  stay within 5% of the commanded step.
```

What is typically desired is a quick rise time with little or no overshoot and quick settling time. The case of 'critical damping' -- the fastest possible rise time that creates no overshoot -- is often the goal. There are usually tradeoffs between these parameters, particularly between fast response and low overshoot.

A FEW NOTES

If your amplifier has a tachometer, the tach is providing derivative gain (and therefore damping) within the amplifier itself. If the amplifier has been well tuned, you should not have to add any more derivative gain in the digital filter, but you are free to do so if you wish.

On the SMCC-P, it is possible to have the error integration active at all times by setting i09 to 0, or to have it active only when the motion is stopped by setting i09 to 1. While the step response for these two cases will look essentially identical, the behavior on real moves will be very different. Error integration that is active at all times can reduce following error on an extended profiled move, but at the cost of reduced system stability and of overshoot at the end of the move (which makes up for the lag at the beginning of the move). In a system without feedforward, the close following may be worth these costs. But the velocity and acceleration feedforward terms on the SMCC-P can virtually eliminate following error without these drawbacks. For this reason, almost all SMCC-P customers use error integration only when motion is stopped -- where it can eliminate steady state errors due to static friction or net torque loads.

Because the proportional gain term Kp is outside the brackets in the filter equation (see above), it also affects derivative and integral gains, and is not strictly speaking a true proportional gain. For this reason, if you modify Kp when Ki or Kd is not equal to zero, you are also changing the effective integral or differential gain. The shape of the response curve will not change much, although its timing will. You will want to change Ki and/or Kd in the opposite direction from Kp if you want to keep their effective gains constant.

Feedforward will affect step response even though it has no effect on the system stability we are really evaluating. Be sure that both acceleration and velocity feedforward are set to zero as you are doing the step responses.

If you are working with an AC induction motor, note that for low filter values, there is an oscillatory mode to the response that you will not see in other motors. As the proportional gain is increased, the magnitude of the oscillation will not increase much, but the frequency will. The decay time of this oscillation decreases in proportion to the increase in frequency, so increasing Kp will cause faster decay of the oscillation. As with other motors, derivative gain is used to damp out this oscillation. At gains high enough for quick response and good stiffness, you may see residual oscillation of a couple of encoder counts. This is to be expected. (See the AC induction step examples, below, near the end.)

The default step size of 50 encoder counts should be adequate for most systems. The guidelines are to make the step large enough so that the granularity of the position measurement is not a nuisance, but small enough so that the filter does not saturate on the step (step size times proportional gain should be less than 500,000 with some margin; for instance a step size of 100 with a proportional gain of 6000 will saturate, giving a misleading response).

Some systems will have mechanical resonances in the coupling of the motor to the load. The PID filter cannot compensate for these resonances; if their presence is not tolerable, you must keep the gains low enough to stimulate them, or (preferably) stiffen the coupling to reduce the resonances. (See the examples of a system with resonance, below, near the end.)

FEEDBACK PROCEDURE

1. Set a 'safe' starting filter with a little proportional gain, with no (or almost no) derivative or integral gain, and no feedforward. The default starting filter (hit F1) provides a value of 400 for Kp (100 if there is a tachometer), 5 for Kd (0 if there is a tach), and 1 for Ki (this is just to make sure that the system does not drift away from the desired area of operation), with i09 set to 1 so the integrator is active only while stopped. If you do not wish to use this default filter, you can set the parameters individually, with F3 (Kp), F4 (Kd), F5 (Ki), Alt-F3 (Kvff), and Alt-F4 (Kaff). Parameter i09 can be set simply by typing either 'i09 0' or 'i09 1'.

2. Do a step (hit F2) and observe the plotted response displayed on the screen, along with the calculated statistics.

3. Adjust (probably increase) Kp (hit F3, then enter the new value) to get the fastest rise time possible without a huge amount of overshoot. Allow more overshoot here than you will in your final response.

4. Once you have a fast response, increase Kd (hit F4, then enter the new value) to bring down the overshoot to the desired value. Note that this will also increase the rise time.

5. You may need to do further tradeoffs between Kp and Kd to get the desired response.

Note: You may wish to change the size and/or the duration of the step to be able to observe the response better. Hit the F6 key, and then follow the directions you are given. The default values are a 50 count step with a 400 millisecond dwell time.

6. Once you have set Kp and Kd, you have taken care of your dynamic step response (provided you are using error integration only in position). Now you will want to add integration to improve the static holding properties of the system. As you increase Ki and observe the step response, you will notice that it increases overshoot but comes back to the command position more quickly. A good value for Ki is one that brings the response back down to the command position as quickly as possible without going back past it.

EXAMPLE

The following figures are taken from the actual tuning procedure for a motor using this program. They were done on a small DC laboratory motor with virtually no load, so the response is faster than it would be in almost any real-world application. The actual times are not important, however; the shapes of the response curves are. This system has a current-controlled amplifier with no tachometer. The goals of this system are critically damped step response, the quick elimination of steady-state errors at rest, and the minimization of following errors.

Figure 1 shows the step response with absolutely no damping added (derivative gain). The large amount of ringing is obviously unacceptable; some Kd is needed. Figure 2 shows the response with a Kd of 5 (this is the default starting filter). The ringing is largely eliminated; there is an overshoot of 36% and a rise time of 14 msec. Let us see if we can make the response quicker (which means a stiffer system).

Figures 3 and 4 show the response with Kp increased to 1000 and 2000, respectively. Note that the shape of the curve has not changed much (this is because the effective derivative gain is increasing with Kp), but the rise time has gone down to 8 and 6 msec, respectively. Since increasing Kp seems to be reaching the point of diminishing returns as far as reducing rise time, we turn now to Kd.

Figure 5 shows the step response with a Kd of 7. This is the critically damped case -- fast response with no overshoot. With Kd any smaller, we get some overshoot; with it any larger, we just slow down the response. The one-count error in the step is due to a net torque or static friction; we will eliminate this with integral gain.

Figure 6 shows what happens with a little bit of integral gain: the steady-state error is gone, but the nature of the curve has not really changed.

Figures 7 and 8 display the response curves for substantially increased Ki (50 and 200 respectively). These curves demonstrate how quickly the system would respond to a disturbance while in position. (Remember that we are using integral gain only when in position, so changing integral gain does not affect our actual dynamic response, although it will change the shape of the step response here. We still have the stability characteristics of critical damping while on the move.) Even the higher value of Ki does not result in oscillations, so we will use that. We have achieved what we wanted with feedback; next we tackle feedforward.

.PA
.PA
Substitute Figures 1 thru 4 for this page.
.PA
.PA

Substitute Figures 5 thru 8 for this page.
.PA

FEEDFORWARD TUNING WITH PROFILED PARABOLIC MOVES

In a position servo system without feedforward or dynamic error integration, there must be a continual error between the commanded position and the actual position in a profiled move (known as following error) to produce a motor command. This means, however, that you are never really where you want to be when you want to be there, which is often the point of a profiled move in the first place. These following errors will usually be proportional (well correlated) to the velocity and/or the acceleration. The velocity and acceleration feedforward terms can be used to reduce these following errors virtually to zero. These parameters add terms to the torque command that are proportional to the commanded velocity and acceleration, respectively, in each cycle of the profiled move.

Mathematically speaking, if two sets of data, such as velocity and following error, vary in complete proportion to each other, they have a correlation of 1.0 (perfect correlation). If they vary completely independently of each other, they have a correlation of 0.0 (no correlation). The more they vary in proportion to each other, the closer their correlation will be to 1.0. In graphical terms, the more two curves are shaped like each other, the better they will be correlated. Another important figure is the constant of proportionality between the two sets of data, which is the average ratio between matching points in the sets. Even if two sets of data are very well correlated, the ratio may be so low that one of the sets is negligible in practical terms.

For each move done, the program will calculate the correlation between velocity and following error, and between acceleration and following error. It will also calculate the average ratio between following error and both velocity and acceleration. As a feedforward gain is increased, the ratio of following error to that quantity will decrease linearly (e.g. if a gain of 0 produces a ratio of 12.0, and a gain of 5 produces a ratio of 6.0, then a gain of 10 can be expected to drive the ratio to zero). The ratio will decline even as the correlation stays high; when the ratio gets small enough, the correlation should decrease as some other factor becomes the dominant cause of following error (e.g. noise or the other feedforward gain). Ideally, the correlation will be brought near zero as well as the ratio, but the resolution in feedforward gain may not allow this (a gain of x may have a strong positive correlation, but a gain of x + 1 has a strong negative correlation -- see the example below). If this is the case, keep the gain that causes a positive correlation: the ratio (and so the magnitude) of following error will not be very large.

The SMCC-P has two feedforward terms:

```
     Kvff -- velocity feedforward gain -- i22 [i42 for y-axis]
     Kaff -- acceleration feedforward gain -- i30 [i50]
```

The strategy in this section is to do a series of 'parabolic' moves (cubic in position, parabolic in velocity, linearly varying in acceleration), while adjusting the feedforward terms to reduce the following error and its correlation to velocity and acceleration. After each move, the program will automatically calculate the correlations and ratios, and the maximum following error. These will be displayed on a plot of the position and the following error. The feedforward terms are increased from zero (working first with velocity feedforward) until the ratios (and hopefully the correlations are as close to zero as possible, without going strongly negative. If either correlation goes very far negative, you will be likely to get overshoot at the end of a move.

A FEW NOTES

To get meaningful correlation information, particularly about acceleration, you must 'push' the system hard. The ALT-F1 key allows you to change the length and duration of the move. By increasing the length and/or decreasing the time of the move, you can get higher velocities and accelerations. Decreasing time is appropriate if increasing the length would cause problems with maximum travel or top velocity.

In a system without a tachometer, you will probably want to set the velocity feedforward equal to the derivative gain. In a system using a tachometer, the velocity feedforward should be greater than the derivative gain.

When you do the parabolic move without any feedforward, you will probably see a very high correlation to velocity (~1.0) and almost no correlation to acceleration (~0.0). There is most likely a real correlation to acceleration, but it is swamped out by the velocity correlation. As you reduce the velocity correlation, you should see increased acceleration correlation. When you then reduce the acceleration correlation, the correlation to velocity may increase again, but the actual ratio and magnitude of the FE should be very small.

Parabolic moves were chosen for this procedure because their acceleration and velocity vary continually and are uncorrelated to each other, making them ideal for this type of analysis. It is also possible to use trapezoidal moves here, or just as a final check. Trapezoidal moves only have three levels of

acceleration, however, and a long period of constant velocity, so the correlation information might not be as good. These moves are performed by hitting Alt-F8, and their parameters are adjusted by hitting Alt-F9.

For further examination of the move, you may plot the velocity curve (hit Alt-F5), the acceleration curve (hit Alt-F6), or the following error curve (hit Alt-F7). These are automatically scaled to fill up the plot window for maximum resolution. The move statistics are re-displayed with these plots.

PARABOLIC RESPONSE/FEEDFORWARD PROCEDURE

1. Do a parabolic move (hit ALT-F2). Observe the plotted response and the calculated statistics for the move.

2. Presuming there is a high correlation between following error and velocity, increase Kvff (hit ALT-F3, then enter the new number).

3. Do another parabolic move. If there is still inadequate Kvff, there will still be a high correlation, but the FE-to-Vel ratio and the maximum FE will be reduced. Repeat steps 2 and 3 until you have the maximum Kvff that produces a positive ratio and correlation value.

4. At this point, there should be noticeable correlation between acceleration and following error. You can see this by the numerical correlation value, or by plotting the acceleration and the following error, and noticing the similarity in shape. If you do not see any correlation, try increasing the length or decreasing the time of the move to get higher accelerations.

5. Increase Kaff (hit ALT-F4, then enter the new number). Do another move, and evaluate the correlation and FE-to-Accel ratio again. Repeat until you have the maximum Kaff that retains any positive correlation. At this point, you should have minimal following error, and most of what remains should be noise.

6. Pop open a beer. You are done!

FEEDFORWARD EXAMPLE

In Figure 9, we see the plot of a parabolic move using the same filter values that we had at the end of the feedback tuning. Both position and following error (FE) are plotted against time, with the statistics on the following error shown. The FE gets as large as 412 counts, and it is virtually perfectly correlated to velocity (Vel. corr. = 1.0), with a ratio of 1.808. It shows essentially no correlation to acceleration (Acc. corr. = 0.041). Figure 10 shows the plot of velocity versus time for the same move; Figure 11 shows acceleration versus time; and Figure 12 shows FE versus time. Note that the velocity and FE curves are shaped almost exactly alike; this is a ramification of their high correlation. Conversely, the acceleration and FE curves, looking nothing alike, have almost no correlation.

Figures 13 and 14 show the position and FE curves with Kvff increased to 5. The maximum FE is well down -- to 120 counts (and a ratio of 0.497), but it is still almost perfectly correlated to velocity (still shaped the same -- refer back to Figure 10). Knowing that Kvff=0 produced a ratio of 1.808 and Kvff=5 produced a ratio of 0.497, we can extrapolate and predict that Kvff=7 will bring the ratio almost exactly to zero. Figure 15 shows the FE with Kvff increased to 7. The ratio has been
.PA
Substitute Figures 9 thru 12 for this page.
.PA

.PA
Substitute Figures 13 thru 16 for this page.
.PA
brought down to 0.027, much as predicted. The correlation to velocity is down to 0.335, but now we see that the correlation to acceleration is high at 0.903. We can see that Figure 15 is shaped a lot more like Figure 11 (acceleration) than it is like Figure 10 (velocity). (Actually, since this is a non-tachometer system, we could have guessed ahead of time that Kvff=7 would be best, since that is equal to Kd, but there is no similar guide for tachometer-based systems.)

Next we increase Kaff to try to get rid of this correlation. In Figure 16 we see the FE for Kaff=10. The curve has turned upside-down, and we see a strong negative correlation (-0.793). We have added too much acceleration feedforward! Using the same technique as above, we note that Kaff=0 produced a ratio of FE to acceleration of 1.214, and Kaff=10 produced a ratio of -0.793. Interpolating, we can calculate that Kaff=6 will bring this ratio as close as possible to zero. In Figure 17, with Kaff=6, the ratio is down to 0.038, and there is no remaining correlation to acceleration. The following error is now extremely small through the entire move (maximum of 3 counts), but a correlation to velocity has reappeared. Can we do any better?

Figure 18 shows the FE when Kvff has been increased to 8. The maximum error is up to 56 counts, with strong negative correlation. Since we are not allowed fractional values for Kvff, 7 is the optimum choice. This is equal to Kd, as expected for a non-tachometer system. We are finished tuning the filter!

Figures 19 through 22 show the plots for a move of trapezoidal velocity profile done with this tuned filter (this is more like the type of move many users will be performing in actual use. The following error is still extremely small through the entire move, and the remaining correlations are much the same as for the parabolic move.

EXAMPLE OF TACHOMETER-BASED SYSTEM WITH RESONANT COUPLING

This set of examples deals with a DC brushed motor driven by a tachometer-based amplifier and connected to its load by a long cable that has noticeable springiness (and therefore resonant modes). Figure 23 shows the step response with the default starting filter for a tachomer-based system. The exponential response is quite slow (overdamped). No resonance is visible.

In Figure 24, Kp has been increased from 100 to 250. A couple of things should be noticed in comparing this plot to Figure 23. First, the gross shape of the curve has changed. For a tachometer-based system, the damping is provided in hardware, not software. Unlike damping provided by the Kd term in the SMCC filter equation, it will not increase in proportion to Kp, so the shape of the curve will change. Second, there are some strange oscillations on the curve. These are obviously not the result of the filter trying to correct for overshoot, because they start

before any overshoot. They are due to resonance in the cable
.PA
Substitute Figures 17 thru 20 for this page.
.PA
.PA
Substitute Figures 21 thru 24 for this page.
.PA

coupling. The user of the system would have to decide if he could deal with this type of oscillation, with the options of physically stiffening the coupling, or weakening the filter so this resonance would not be excited.

AC INDUCTION EXAMPLE

As noted above, the commutated AC induction motor can have different characteristics from DC motors when the gain is below what is ideal (it does not behave much like a second-order system in this case). Figure 25 shows the step response of a 5 horsepower AC indcution motor with the filter gains at the default starting values. There is a lot of ringing, and it will obviously take a long time to settle.

In Figure 26, Kd has been doubled, from 5 to 10. The ringing has been decreased a little, but it is still at the same frequency, and not decaying quickly. Next, in Figure 27, Kp has been increased from 400 to 1000. The overshoot has not increased substantially, but the oscillation frequency is higher, and it decays much more quickly. In Figure 28, Kp has been increased to 3000. The oscillations very quickly die out to a magnitude of 1 or 2 counts. There is still some overshoot, which could be eliminated with increased Kd, if desired.

FUNCTION KEY SUMMARY

<F1> -- Send Starting Filter -- Sends default starting filter parameters for the current axis. The intent is to provide a weak but safe filter to start the analysis.

<F2> -- Do A Step -- Performs a step move on the current axis, capturing and plotting the data with key statistics.

<F3> -- Set Proportional Gain -- Lets the user enter a new value for Kp, the proportional gain, for the current axis and card.

<F4> -- Set Derivative Gain -- Lets the user enter a new value for Kd, the derivative gain, for the current axis and card.

<F5> -- Set Integral Gain -- Lets the user enter a new value for Ki, the integral gain, for the current axis and card.

<F6> -- Set Step Parameters -- Lets the user enter new values for either the size of the step or its duration.

<F7> -- Set Induction Parameters -- For AC induction motors only.

Lets the user enter new values for either the magnetization current or the slip gain of the AC induction motor.

<F8> -- Change Axis -- Changes the axis to be worked with (if the card is configured for two axes).
.PA
Substitute Figures 25 thru 28 for this page.
.PA

<F9> -- Starting Help -- Explains the basic instructions for this section of the program.

<F10> -- Feedback Help -- Explains the procedure for adjusting the feedback gains using step response.

<Alt-F1> -- Set Parabolic Parameters -- Lets the user enter new values for either the length or the duration of the parabolic move.

<Alt-F2> -- Do a Parabolic Move -- Performs a parabolic move on the current axis, capturing and plotting the data with key statistics.

<Alt-F3> -- Set Velocity Feedforward -- Lets the user enter a new value for Kvff, velocity feedforward gain, for the current axis and card.

<Alt-F4> -- Set Acceleration Feedforward -- Lets the user enter a new value for Kaff, acceleration feedforward gain, for the current axis and card.

<Alt-F5> -- Plot Velocity -- Draws a graph of velocity versus time for the most recent profiled move, with key statistics.

<Alt-F6> -- Plot Acceleration -- Draws a graph of acceleration versus time for the most recent profiled move, with key statistics.

<Alt-F7> -- Plot Following Error -- Draws a graph of following error versus time for the most recent profiled move, with key statistics.

<Alt-F8> -- Do A Trapezoidal Move -- Performs a trapezoidal move on the current axis, capturing and plotting the data with key statistics.

<Alt-F9> -- Set Trapezoidal Parameters -- Lets the user enter new values for either length, duration, or acceleration time of the trapezoidal move.

<Alt-F10> -- Feedforward Help -- Explains the procedure for tuning the feedforward parameters using profiled moves.

.pa
6.0 SMCC DATA GATHERER/PLOTTER/PRINTER SECTION

This section allows an SMCC card with Parabolic option (SMCC-P, option 4) to collect data at servo cycle rates for performance analysis (i.e. it can generate a "snapshot" of position, velocity, acceleration, following error, or other data). The data is brought back from the card into the computer, where it can be stored to disk, displayed or printed in raw or formatted form, or graphed.

Once you have entered this section by pressing <F10> in the SMCC Terminal main menu, you are presented with the SMCC Plotter screen. Typing alphanumeric keys still lets the left window work like a standard terminal. In the right window you see the sub-menu of function key operations for this section. The menu appears as:

```
     F1 - Collect Data
     F2 - Collect/Store
     F3 - Get Disk Data
     F4 - Plot Data
     F5 - Disp Raw Data
     F6 - Disp Dec Data
     F7 - Print Raw Data
     F8 - Print Dec Data
     F9 - Sys Command
     F10 - Plot Help
```

The instructions for each of these function keys are detailed below:

<F1> - Collect Data

Pressing F1 causes the SMCC program to check the card you are connected to and ensure that the card is one with the parabolic option (Opt. 4). If it is not, you may not proceed any further. If it is a parabolic card, the program will ask you the period of data collection you wish to use. Your choices are in numbers of servo cycles (the default value for a servo cycle is 976 microseconds). Your choices are displayed as:

```
          <0>   1  <1>   2  <2>   4  <3>   8
          <4>  16  <5>  32  <6>  64  <7> 128
```

Choose the period you want by pressing your selection of 0 to 7. For example, choosing <4> would give you a period of 32 servo cycles, or 32 x 0.976 = 31.232 milliseconds per sample.

Next, the program asks you to select the type of data you wish to collect. Your choices are displayed as:

```
          <1>  X Pos.   <2>  X & Y Pos.   <3> X Pos. & FE
          <4>  (X & Y) (Pos. & FE)        <5> Special Data
          (Vel. & Accel. data computed from position)
```

Enter your choice by pressing your selection of 1 to 5. If you chose one of the standard selections (n = 1 to 4), there will be 4 x n bytes collected each sample, since each value is a 4-byte number. As noted in the display of the choices, if position data is acquired for an axis, the program will automatically compute velocity and acceleration data (using forward-backward difference).

If you press <5> for special data gathering, the program will ask you for the number of bytes to store each sample. You can store from 1 to 32 contiguous bytes each sample. Next, you will be asked for the capture address, which is the address of the first byte to be collected (in the SMCC processor's memory space).

At this point, you are ready to run a program in SMCC and have it collect data for you while it runs the program. The SMCC Terminal program tells you to command the card in terminal mode to perform your desired actions. When you are done performing the moves and storing data, you type CTRL-Z, and the data will be brought back into the computer. The program in the card must set and clear 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)
```

For informational purposes, there follows here a brief explanation of how the data gathering works. The SMCC-P command to set up 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 SMCC-P will prepare for data storage and return the address of the pointer to the data storage buffer. Before any storage, this will be the beginning of the buffer. (A 'y' command sent without any arguments will do nothing but return the address of the pointer. This can be used after gathering the data to find the end of the buffer used, and therefore, the length of the buffer.)

The SMCC terminal program automatically sends this command with parameters based on your entries to prepare for storage. When the program on the card is run, data storage will begin when the storage bit is set, and it will end when the storage bit is cleared (or when the data storage buffer is filled). When you type CTRL-Z, the computer sends the 'y' command without arguments to find the end address of the data storage, and brings the data back with the 'd' command.

The amount of storage space the SMCC-P has for data gethering depends on the size of your program buffer. For example, if you have a one hundred line program with no subroutines and no background programs, you will have approximately 6K of data storage space, so that if the data gathered is 16 bytes every 128 milliseconds, the SMCC will run out of room in (6000/16) * 0.128 seconds = 48 seconds (approximately). When SMCC's Option 5 (32K memory) is used, about 4 times as much can be stored.

<F2> - Collect/Store

This operates identically to F1, except that it also stores the data in a disk file. You will be asked for the name of the disk file before you start the data collection.

<F3> - Get Disk Data

This section allows to you to bring in previously collected data that had been stored on disk (see instructions under F2 and F1). You will be asked for the name of the disk file from which you wish to retrieve the data. Once the data has been brought into the computer, you may use it just as if it had been brought in directly from the card. You do not need to be connected to a card to get the data in this manner.

<F4> - Plot Data

This section lets you view the collected data in graphical form. You must have graphics capabilities on your PC to utilize this function. You have a great deal of choice in what data you plot, and how you plot it. You will be presented a list of choices in the left window:

```
<0> Change Plot Settings
<1> X Position vs Time
<2> X Velocity vs Time
<3> X Acceleration vs Time
<4> X Following Error vs Time
<5> Y Position vs Time
<6> Y Velocity vs Time
<7> Y Acceleration vs Time
<8> Y Following Error vs Time
```

```
     <9> Special Plots
     Enter your choice (ESC to exit):
```

Items <4> through <8> will only be displayed if the data for that choice has been collected.

Selecting an item <1> through <8> will cause that plot to be displayed on the screen, replacing the text screen you were using. Once in the graphics screen, typing 'P' or 'p' will cause the plot to be copied to a printer (which must bean Epson-compatible graphics-capable printer in the standard printer port). Hitting any other key will cause a return to the text screen.

If you select <0>, you will be presented with a series of questions for setting variables that control how the data is plotted, including what units will be used (which affects the tabular output as well -- see <F6> and <F8>), where the legend is located, and whether the commanded or actual data is shown.

First, you will be presented with the selection of units for time, looking much as below:

```
       Time units:
     <0> (User defined)
     <1> Servo Cycs
     <2> Sample Pds
     <3> Seconds
     <4> Minutes
     Current setting is <3>: Seconds.
     Enter you choice (return to keep):
```

Here you enter a number from 0 through 4 to make your selection. If you choose <0> to define your own time units, you will be asked for the name of your units, and then for the number of seconds in one of your units (e.g. you could define millisecs as having .001 seconds).

Following this, you will be presented with similar selection options for position units, velocity units, and acceleration units, where the choices work identically.

Next you will be asked if you wish to show the actual data read from the encoder, or the commanded data created by the trajectory generator. You must have following error data to have the choice of command data (actual position + following error = command position). If you have set this variable to show command data, and you then bring in data without following error, the variable will automatically be set back to show actual data.

After this, you will be asked if you want to display a graticule superimposed on your plot (for aid in measurement), then in which corner of the plot you wish to show the legend. Finally, you will be returned to the Plot Options menu.

If you select item <9> from the Plot Options menu, you can build your own plot from the variables available. You will be presented with a list of these variables, some or all of the following, depending on what has been brought into the computer:

```
<0> Time
<1> X Position
<2> X Velocity
<3> X Acceleration
<4> X Following Error
<5> Y Position
<6> Y Velocity
<7> Y Acceleration
<8> Y Following Error
<9> Last Special Plot
```

First, you will be asked which of these variables you wish to plot on the horizontal axis (all of the standard plots use time). Indicate your choice by entering the number of the choice. Second, the program will ask you which variable you wish to plot on the vertical axis (time is not available as a choice on the vertical axis). Enter the number of your choice. Entering <9> will give you the last special plot you defined, without any further questions.

Next it will ask you if you wish to plot a second variable vertically against the horizontal variable. If you answer positively ('Y' or 'y') it will ask you what the second vertical variable is, and again you enter the number of your choice.

If your two vertical variables are of the same type of unit (e.g. both velocity, or position and following error), you will be asked how you wish to do this double plot. The choices are:

```
<0> Plot curves separately, same scale
<1> Plot curves separately, diff. scales
<2> Plot sum of curves
<3> Plot difference of curves
```

Choosing <0> will produce a plot with two curves plotted with respect to the same vertical scale. Choosing <1> will show two curves, the first plotted with respect to the left vertical scale, the second with respect to the right vertical scale. A choice of <2> will yield a single curve that is the sum of the two vertical variables. A choice of <3> will show one curve with the second variable subtracted from the first.

If the two vertical variables are of different types of units, they will automatically be plotted as separate curves on different scales.

If you have collected data using the SPECIAL DATA option, and the variables were four bytes each, you may be able to use this routine profitably, although the titles will be misleading. You may have to experiment with the unit scaling to get the output you desire.

<F5> - Display Raw Data

This key lets you see the collected data in 'raw' form, that is, just as it was gathered from the card, before it was separated into variables and scaled. For some SPECIAL DATA sets, this may be the only meaningful way to examine the data. It is shown in hexadecimal form with a line looking like this:

```
     a4b4 e0 c8 13 00 20 00 00 00 80 c7 13 00 40 00 00 00
```

The first four digits show the starting address of the line in the data storage buffer of the card. Each pair of digits following is the hexadecimal representation of a byte of memory. For the standard collected variables of position and following error, these are grouped in sets of four bytes (least significant byte first) to form a 32-bit value.

NOTE: This routine must re-access the card or disk from which the data came, so the source must be available to the program throughout the routine.

The data is displayed one screen (twenty-three lines) at a time in the left window. At the bottom of the right window are shown the keys that allow you to move between screens of data. The keys are:

PAGE DOWN - Show the next page of data
PAGE UP - Show the previous page of data
END - Show the last page of data
HOME - Show the first page of data
ESC - Exit this display routine
(Any other key) - Same as PAGE DOWN

<F6> - Display Dec Data

This key allows you to view the data on the screen in formatted and scaled form. Since this often requires the full screen width, it will be displayed without the standard frame. First, a page of instructions will be displayed in the left window. When you proceed, the data will be shown one screen (twenty-three samples, or lines) at a time, with a title row and a units row at the top of the screen. The keys to control movement between screens of data are the same as those for the display of raw data, but there will not be a menu continually shown, because the data takes up the full screen.

The units and scaling of the variables are the same as for plotting. These can be changed in the 'Plot Data' section (see <F4>, option <0> above).

If you have collected data using the SPECIAL DATA option, and the variables were four bytes each, you may be able to use this routine profitably, although the titles will be misleading. You may have to experiment with the unit scaling to get the output you desire.

<F7> - Print Raw Data

This key allows you to print the data you have collected in its original unformatted form to an ASCII printer at the standard printer port. The printout can be aborted by hitting the ESC key during the print. For more details see <F5>, above.

NOTE: This routine must re-access the card or disk from which the data came, so the source must be available to the program throughout the routine.

<F8> - Print Dec Data

This key allows you to print the data you have collected in formatted and scaled form to an ASCII printer at the standard printer port. The printout can be aborted by hitting the ESC key during the print. The units and scaling are governed the same way as in the display of formatted data (see <F6> directions, above).

<F9> - Sys Commands

This key allows you to perform DOS system commands without actually exiting the program. It is particularly useful here to examine directories for disk files. You can return to the program by hitting <ENTER> or <ESC> at the command prompt.

<F10> - Plot Help

This key lets you view a help screen that is a brief summary of the instructions in these pages.

.pa
7.0 NOTES

For RS232 - As a Summary, the following information pertains to RS232 Communication:

```
     SMCC's RS232 data Format is:  1 Start Bit
                                   8 Data Bits
                                   1 Stop Bit
                                   No Parity.
```

Please refer to attached drawing for correct wiring of the SMCC to the host computer.

Please note that the SMCC Card has an etched trace jumpering DSR to DTR since the software does not support these lines. Also, if there is a need to switch TXD/RXD and RTS/CTS around for Data Terminal/Data Set requirements the SMCC Card has two pairs of selectable jumpers, "E7" and "E8" respectively, to help you do this easily.

Under normal circumstances, you should not have to make any changes. The RS232 Cable provided as "Accessory 3" is 10 feet long and is used to interconnect the SMCC Card to your host computer. General rules on communication distance are:

```
38400 Baud    4 ft.
19200 Baud    8 ft.
 9600 Baud   16 ft.
     !         !
     !         !
     !         !
Double the distance each time Baud Rate is
reduced by half.
```

Shielding the longer RS232 Cables is highly recommended. It is also advised not to run the RS232 Cable next to high power or A.C. signals such as the servo Amplifier, line voltage etc.

FOR PARALLEL - Refer to the "Accessory" 10 IBM P.C. Bidirectional I/O Card Manual.

DEMO FILES ON THE SUPPLIED DISKETTE

The following files are available on the supplied diskette and may be loaded with the SMCC Editor:

```
DEMOXY1.SMC - Simple two-axis demonstration program.
DEMOXY2.SMC - More complex two-axis demo program.
DEMODAT.SMC - Sample program for data gathering.
DEMODAT.PLT - Plot Data
DEMO1.SMC   - Subroutine Demo
DEMO2.SMC   - GOT DEMO
DEMO3.SMC   - S Curve Generator
```

.PA
NOTE FOR IBM AT USERS:

The "X" Clock speed setting SMCC asks about during the boot procedure has a range of X1 to X9 and is used to primarily control communication delays between SMCC and the Smart Motion Control Card. The larger the number, the greater the delay. The faster your computer is, the greater the X number has to be in order to provide adequate delays. In general, recommended settings are:

X1	IBM P.C. at 4 MHz.
X2	Turbo P.C.
X3/4	IBM P.C. at 8 MHz.
X5	Sys. 2

Use the minimum X number that will result in reliable bidirectional communications, using too big a number will unnecessarily slow communications down. Using too small a number can cause character drop-outs.

USE OF DUMP PROGRAM:

The Dump program is also written in "C" language, and is specifically written to allow the user to download files that are

larger than the buffer size of the SMCC.

The Dump program is invoked by typing DUMP, followed by the Filename.Extension of the file you wish to execute. The DUMP program executes a z1000, reserving 1000 program lines, (Therefore, do not use a z command in the file to be dumped.) and then proceeds to download the program 500 lines at a time, by examining the "PB" and "PC" command responses (program pointers), until the entire program has been executed. Dump keeps track of program execution until the entire file is downloaded and executed. SMCC does not have to be stopped at any time during this process, therefore very large files can be executed continuously. The main purpose of this program is to execute files generated by the SMCCAD Program (Design CAD based shape generator program, using HPGL output) since these files tend to become large. See SMCCAD write-up for more detail.

c:smccv36