IB-11C001

MOTION CONTROL SYSTEMS, MSC SERIES **JUNE 1995**

MACROPROGRAM DEVELOPMENT SYSTEM

INSTRUCTION MANUAL

INDUSTRIAL INDEXING SYSTEMS, Inc.

Revision - E

Approved By:

TABLE OF CONTENTS

1.0 INTRODUCTION

This document is part of a series of books that support Industrial Indexing Systems MSC family of motion control systems. It provides information about the Macroprogram Development System which serves as a tool to assist the user in development of motion control programs.

1.1 OVERVIEW

The Macroprogram Development System is a software tool designed to provide an effective environment for creating Macroprograms for the MSC family of motion controllers. Program development for the MSC consists of creating and editing text files containing the appropriate program instructions, compiling these files to generate executable programs, and on-line program debugging. In addition to these features, the Macroprogram Development System provides aids for disk file maintenance and configuration.

Highlights of the Macroprogram Development System are:

1. Simple entry and editing of programs.

2. Interaction between editing and compiling to quickly identify program lines containing errors.

3. On-line manual describing the purpose and format of each Macroprogramming Language instruction.

4. File manager to simplify creating and deleting Macroprogram source files.

5. Comprehensive real time test and debug facility, including:
   a. Program tracing
   b .Inspection and modification of data values and input/outputs
   c. Monitoring of Controller status.

The remaining sections of this book assume that you are familiar with the operation of your personal computer. If you are not, please refer to your computer documentation.

1.2 MANUAL CONVENTIONS

Throughout this manual, the following typeface conventions are used:

1. Instruction names appear in **bold** print.

2. Optional parameters appear in *italics*.

3. MSDOS commands typed by the user appear in **BOLD**, all capitals.

2.0 INSTALLATION

2.1 HARDWARE CONFIGURATION

The Macroprogram Development System (Toolkit) is designed to be used with an IBM compatible personal computer.

The Toolkit, like any IIS product, has its' own part number. Software part numbers are assigned using the format **SFO-NNNN RX** where **SFO-NNNN** indicates the software part number and **RX** indicates the revision. Whenever the Toolkit is significantly changed, a new part number is assigned. There are several versions of the Toolkit currently in use.

The minimum recommended hardware configuration necessary for using the Macroprogram Development System, part numbers SFO-3040, SFO-3076, SFO-3082 or SFO-3110 is listed below:

- Computer:IBM PC, XT, or AT computer, or compatible
- Disk Storage:Dual 720K disk drives or single 720K disk drive with hard disk
- Display:Monochrome, CGA, EGA or Hercules Display
- Memory:Minimum 512K bytes user memory
- Comm Ports:Asynchronous communications adapter(RS232C serial interface)
- Op Sys:MS-DOS revision 2.0 or later

The Macroprogram Development System, part number SFO-3136, was released primarily for the support of the MSC-850/32 Controller. Since this controller allows for programs up to 64,000 bytes **and** with additional data storage of 64,000 bytes, the Toolkit memory requirements increased dramatically. The minimum recommended hardware configuration necessary for using this version of the Toolkit is as follows:

- Computer:IBM PC AT computer with 286, 386 OR 486 processor, or compatible
- Disk Storage:Dual 720K disk drives or single 720K disk drive with hard disk
- Display:Monochrome, CGA, EGA or Hercules Display
- Memory:Minimum 640K bytes of base user memory and an additional 2MB of extended and/or expanded memory
- Comm Ports:Asynchronous communications adapter(RS232C serial interface)
- Op Sys:MS-DOS revision 5.0 or later
- Clock rate: 16 Mhz or higher

NOTE

Not all PC systems advertised as being "IBM PC compatible" are 100% compatible. The Toolkit may not work on systems which are not truly 100% IBM PC compatible.

2.2 INSTALLATION

The Toolkit is normally supplied on a 3.5", 720 Kbyte diskette. The diskette will be identified with an SFO (System Functional Operation) number which identifies the Toolkit and the current revision level.

2.2.1 DUAL FLOPPY DISK SYSTEM

To install the Toolkit on a dual floppy system, perform the following steps:

1. Obtain two blank, formatted diskettes. One diskette will be used to make a working copy of the IIS Toolkit diskette. The second diskette will serve to store the programs you will develop.
2. Power up your computer and load the MSDOS operating system.
3. When the **A>** prompt appears on the screen, place one of the blank diskettes in the **B** diskette drive, and place the Toolkit diskette in the **A** drive.
4. Type the following command and press the **Enter** key.

   COPY A:*.* B:/V

   This command instructs MSDOS to copy the Toolkit diskette from drive A to drive B and to verify that all information was copied correctly.
5. Remove the IIS Toolkit diskette from drive A and store it in a safe place. Remove the copy from drive B and label it with the appropriate SFO number.
6. Place the working diskette in drive A and the blank data diskette in drive B.
7. Type **IIS** and press the **Enter** key.
8. After a few seconds, the initial Toolkit screen should appear. Proceed to the Configuration section of this chapter for further instructions.

2.2.2 HARD DISK SYSTEM

To install the Toolkit on a hard disk system, perform the following steps. These instructions assume that your hard disk is designated as drive **C**. If your hard disk is designated by a different letter, use that letter in place of **C** in the instructions below.

1. Power up your computer and load the MSDOS operating system.

2. It is recommended that the Toolkit be installed in a subdirectory rather than in the root directory of your disk. Create a subdirectory named **TOOLKIT** by typing the following command and pressing the **Enter** key.

   MKDIR \TOOLKIT

3. It is suggested that you create a subdirectory to contain the programs you will develop using the Toolkit. For example, to create a subdirectory called **MACROS**, type the following command, followed by the **Enter** key:

   MKDIR \MACROS

4. Place the Toolkit diskette in floppy drive **A**.

5. To copy the Toolkit programs to your hard drive, type the following command, followed by the **Enter** key:

   COPY A:*.* \TOOLKIT /V

This command instructs MSDOS to copy the Toolkit diskette from drive A to the \TOOLKIT subdirectory on your hard drive and to verify that all information was copied correctly.

6. Log on to the TOOLKIT directory by typing the following command and then pressing the **Enter** key.

CD \TOOLKIT

7. Type **IIS** and press the **Enter** key.

8. After a few seconds, the initial Toolkit screen should appear. Proceed to the Configuration section of this chapter for further instructions.

9. For subsequent use of the Toolkit, repeat steps 6 and 7.

2.3 CONFIGURATION

If you are using the Toolkit with part number SFO-3136, a "banner" similar to the one below will be displayed for a few seconds, on Toolkit startup:

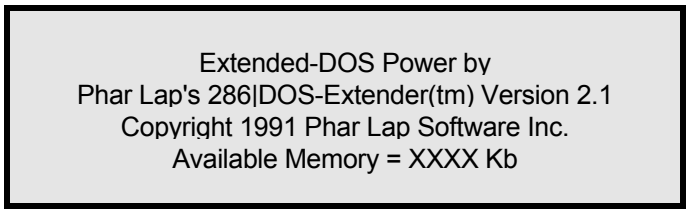

Figure 2.1 - BANNER SCREEN

A software "shell" package is used by the Toolkit in order to access the extended and/or expanded memory in your computer. The "banner" is displayed by this "shell" software package, indicating the manufacturers name, the software "shell" package name, revision and available memory.

If you are using a Toolkit with part number SFO-3110 or SFO-3136, you may be prompted to select the "memory model" to be used for the development of your programs. A screen similar to **Figure 2.2** may be displayed.

The first configuration screen allows you to select the appropriate macroprogram memory size based on your MSC system. See **Figure 2.2**.

STANDARD will accommodate any MSC system as long as your macroprogram is smaller than 16K bytes (PROGRAM + DATA).

MEDIUM would be selected if you are using a MSC-250 and your macroprogram is larger than 16K bytes program + data. The program may not exceed 32K bytes (PROGRAM + DATA).

LARGE is used for the MSC-850/32 system when your macroprogram size exceeds 16K bytes program + data. The program may not exceed either 64K Program and 64K Data.

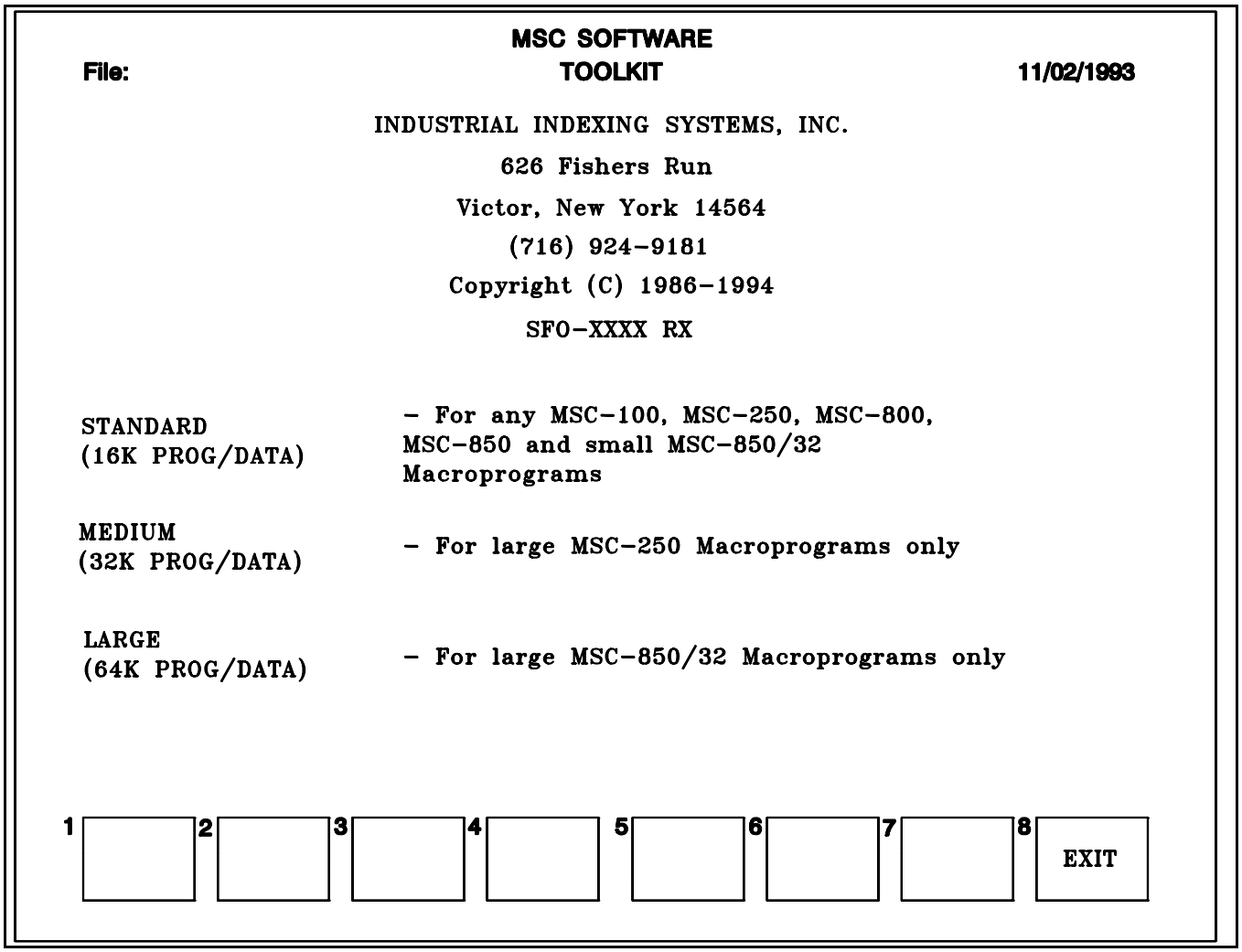

Figure 2.2 - MSC MEMORY SELECTION SCREEN

Generally speaking, if you are creating programs for use with the MSC-850/32 and the program and or data is expected to exceed 16K bytes, you should select the LARGE memory model.

If you are creating programs for use with any other MSC type controller, the STANDARD memory model should be selected, since the sum of the program and data areas cannot exceed 16K bytes anyway.

The Toolkit provides a configuration subsystem that allows you to tailor the Toolkit to your computer. There are two steps to the configuration process. System Configuration covers various options such as specifying where your program files are to be stored, maximum program size in lines, etc. Color Configuration allows you to customize screen colors and characteristics.

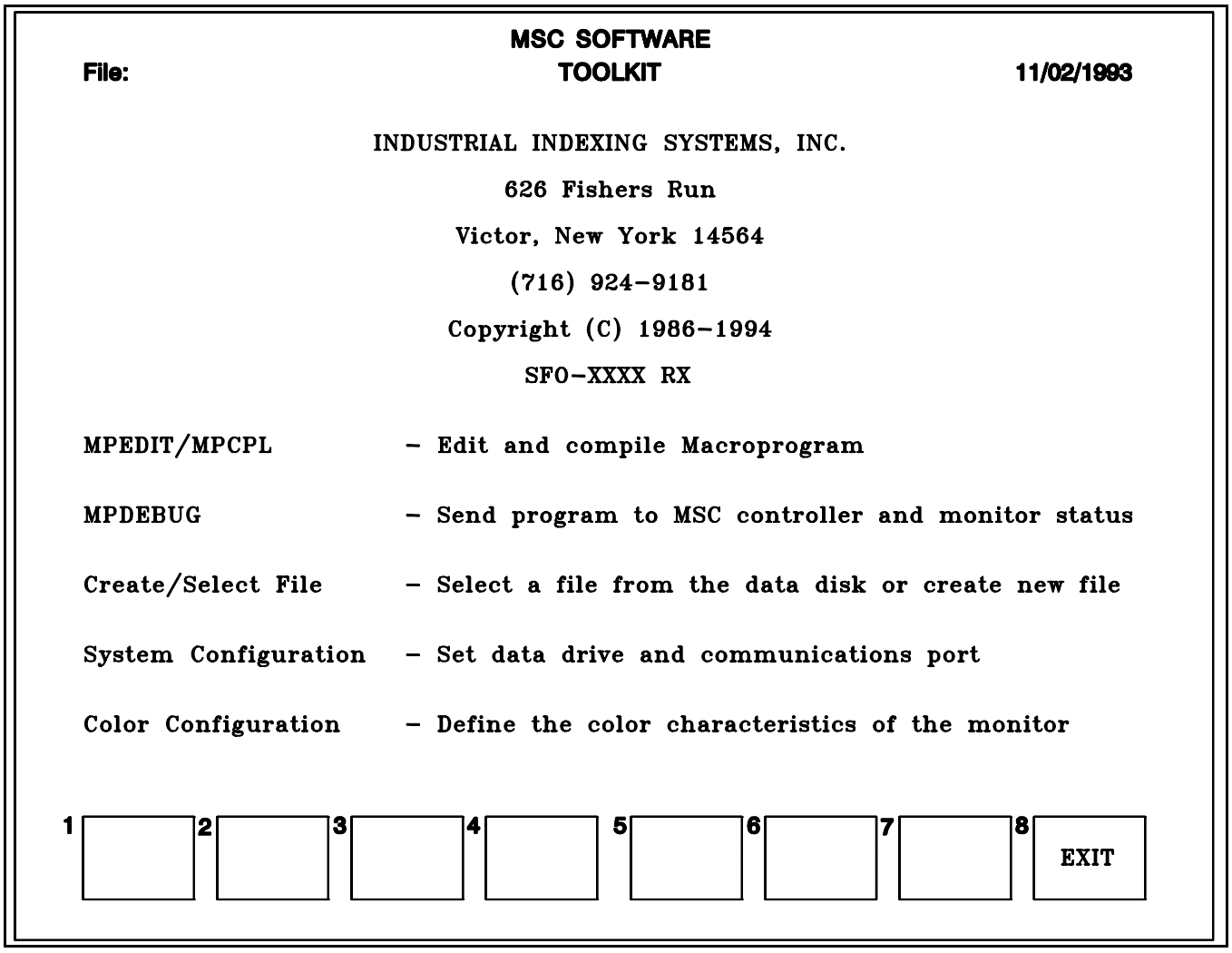

Figure 2.3 - TOOLKIT STARTUP SCREEN

Figure 2.3 shows the Toolkit startup screen. Note that the first selection on the screen, **MPEDIT/MPCPL**, will be highlighted by an inverse video bar. This bar can be moved from one selection to another by using the up and down arrow keys. To activate the desired selection, press the **Enter** key.

2.3.1 SYSTEM CONFIGURATION

The System Configuration option of the Toolkit allows you to set up certain Toolkit default characteristics. These characteristics are usually only set up once, and the configuration will not need to be selected again unless the characteristics change. The System Configuration screen is shown in **Figure 2.4**.

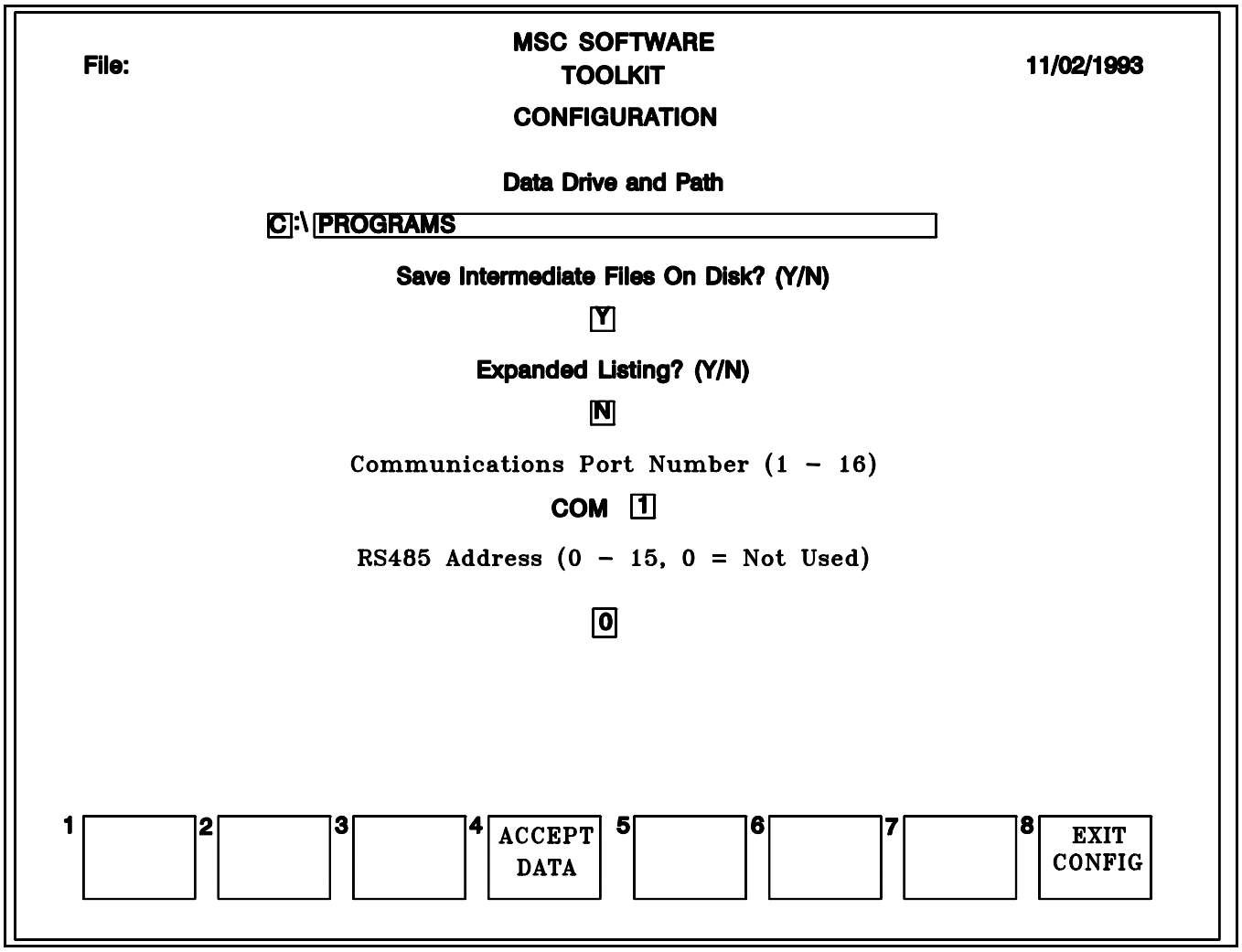

Figure 2.4 - SYSTEM CONFIGURATION SCREEN

The configuration characteristics are as follows:

1. Disk, directory and path selection for program file storage and retrieval.

   The disk selected must be a valid disk on your computer system. Directory and path selection are optional, but if chosen, they too must be valid directories on your computer system. Refer to your computer handbooks or manual for more information on the valid disk and directory names for your computer.

2. Save Intermediate Files on Disk (Y/N)

   During program compilation, the Toolkit creates a number of temporary files which are used during program debugging. If this option is set to **N**, these temporary files are kept only in memory. This speeds up the compilation process. However, if you exit the Toolkit, and at a later time, wish to further test your Macroprogram, the compilation process must be repeated to recreate the temporary files. Setting the option to **Y** causes the temporary files to be written to disk, as well as kept in memory.

3. Expanded Listing (Y/N)

   During program compilation, the development system will, if requested, produce a detailed program listing file. This listing includes tables for the following: all constants, all data variables, all equates, and all instruction labels. The listing also shows all the source program instructions and their compiled "machine" codes. This listing is not required for successful program execution and considerable savings in compilation time and disk storage space can be realized if the listing is omitted.

4. Communications Port Number (1 - 16)

   Many computers have more than one serial communications port available. You may select the appropriate port for your system by entering the appropriate port number. The Toolkit will automatically configure the communications port to the proper settings. Refer to your computer handbook or manual for information on serial communications with your computer. If you are using a port other than 1 or 2, please contact IIS for an application note on non-standard communications ports.

5. RS485 Address (0 - 15)

   If you are using RS485 multi-drop communications, enter the device address that was set up on the MSC controller card. Refer to the instruction book for the type of MSC controller being used.

You can move from field to field on the CONFIGURATION screen by using the **Enter** key and/or the **TAB** and **SHIFT/TAB** keys. Once you have made the appropriate entries, you can press the ACCEPT DATA softkey and these selections will be recorded onto your disk. If you decide not to change these selections, you can press the EXIT CONFIG softkey and the system will return to the Macroprogram Development system selection menu.

2.3.2 COLOR CONFIGURATION

The Color Configuration option of the Toolkit allows the user to customize the colors and/or video enhancements displayed on the computer screen. Selecting this option shows the form displayed in Figure 2.5. The left side of the screen will display eight lines of **A**'s, illustrating the various enhancements and colors that your system is capable of displaying. The right side of the screen represents a miniature Toolkit screen. As you choose various display options, this side of the screen will change accordingly.

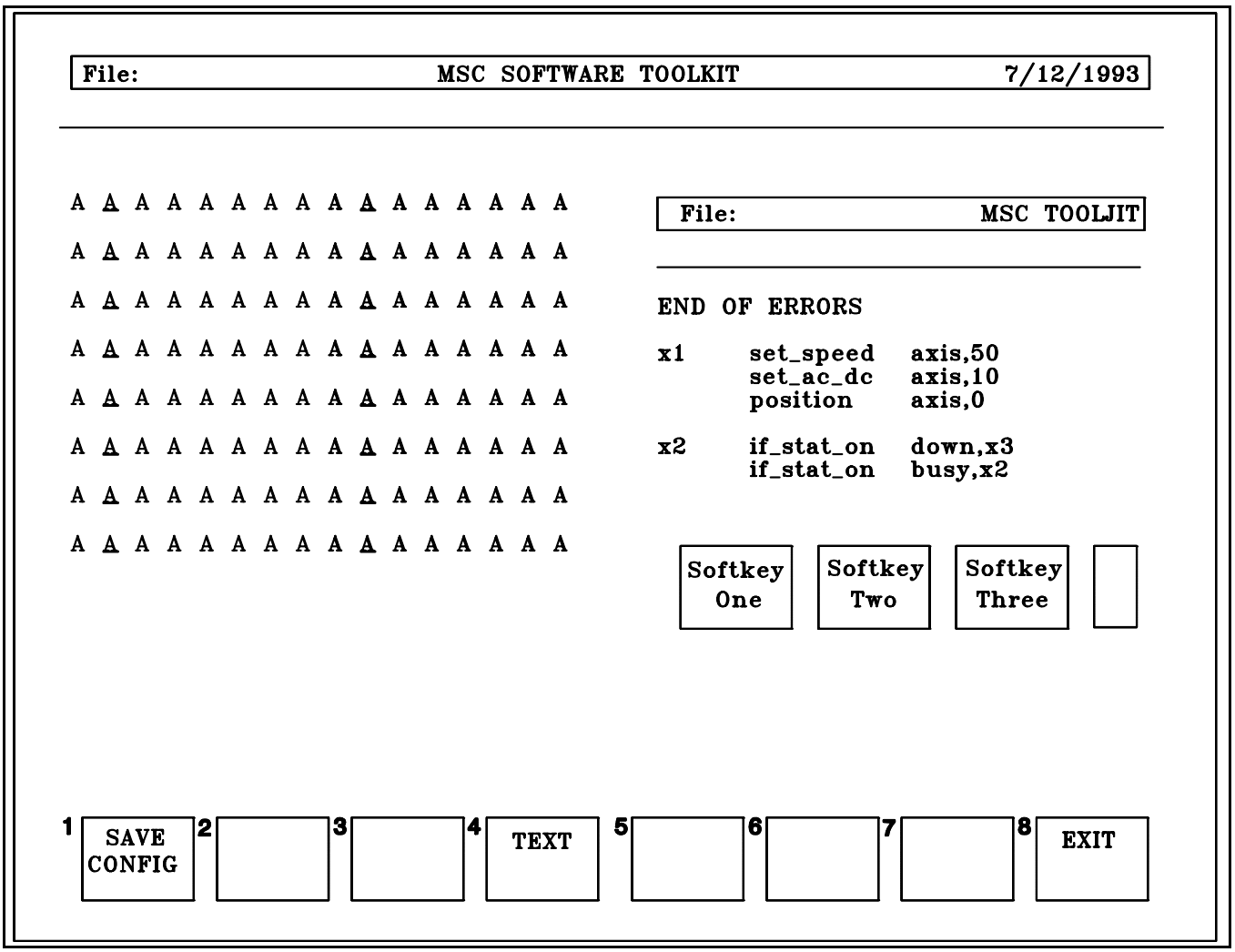

Figure 2.5 - COLOR CONFIGURATION SCREEN

There are six different options which can be set using Color Configuration:

TEXT Controls the enhancement for lines 4 through 21 of the screen.

FUNCTION KEYS Controls the enhancement for the function key labels displayed at the bottom of the screen.

LINE 1 Controls the enhancement for the top line of the screen. This line normally contains the name of the program being developed, the Toolkit banner, and the date.

LINE 2 Controls the enhancement for the second line of the screen. This line is normally used for system messages and user input.

LINE 3 Controls the enhancement for the third line of the screen. This line is normally used for system messages and user input.

CURSOR Controls the enhancement for the higlight bar which appears on certain screens and is moved about using the cursor arrow keys.

To change the enhancement for a particular item, press function key F4 until its label displays the option name. Notice that, on the left side of the screen, a small arrow appears over the **A** which has the current enhancement. Use the cursor arrow keys to move the small arrow to the desired enhancement. Repeat this step for each change you wish to make. When you are satisfied with your choices, press **F1**, **SAVE CONFIG**. Your changes will then be recorded.

WARNING

It is possible to change enhancements so that some or all of the Toolkit screen will not be displayed.

3.0 MPEDIT - MACRO PROGRAM EDITOR

3.1 INTRODUCTION

MPEDIT is a full screen program editor which provides the ability to enter Macroprograms into computer memory and to make modifications simply and effectively.

3.2 FILE MAINTENANCE

To use MPEDIT, it is necessary either to select a currently existing program file, or choose to create a new one. These functions are performed by selecting the **Create/Select File** function from the main menu of the Toolkit. This function will cause a list of any Macroprograms resident on the current data path to be displayed. At this point, you may:

1. Choose an existing Macroprogram file by moving the highlight bar to the desired program and pressing the **Enter** key.
2. Create a new Macroprogram by pressing F4. The Toolkit then asks for the desired file name. Type the file name, followed by the **Enter** key. File names can consists of letters and numbers. Do not use blank spaces in file names. MPEDIT automatically assigns the **.prg** extension to the file name you choose.
3. Delete an existing Macroprogram file by moving the highlight bar to the desired program and pressing F2. The system will prompt you to be sure you wish to delete the file. Answer by pressing the **Y** key or the **N** key accordingly.

After you create or select a file for editing, the Toolkit takes a few seconds to create or read the file and then returns to the main Toolkit menu. At this point, you may choose the MPEDIT option from the menu and press the **Enter** key.

3.3 THE EDITING PROCESS

Entering Macroprograms with MPEDIT is much like using the computer keyboard as a typewriter. Just type the program line as you would like it to appear, and press the **Enter** key when each line is complete. Note that as you type the line, it appears in the highlight bar on your screen. When the return key is pressed, the highlight bar moves down to the next line and the line you typed appears normally. Entering new lines or making corrections always occurs within the highlight bar.

Within this manual, the location of the highlight bar will be referred to as the current line. Within the current line, there will be a blinking underline character, referred to as the cursor.

When typing a line, corrections can be made by backspacing until the cursor is under the error, and retyping the correct information. When making corrections, it is NOT necessary to press the **Enter** key after the correction.

3.4 SPECIAL KEYBOARD KEYS

Table 3.1 - SPECIAL FUNCTION KEY FEATURES

KEY	FUNCTION
Down Arrow	moves current line bar down one line
PgDn	display next page of text
Up Arrow	moves current line bar up one line
PgUp	display previous page of text
Right Arrow	moves cursor one space to the right
Left Arrow	moves cursor one space to the left
SHIFT/DELETE	deletes line at the cursor
SHIFT/INSERT I	nserts a blank line
HOME	moves to the top of the program
END	moves to the bottom of the program
INSERT	turns on insert character mode. Characters subsequently typed will be inserted at the cursor position. Press INSERT again to turn off insert mode
DEL	the character at the current cursor location is deleted
F9	changes from one set of function key definitions to another
TAB	moves the cursor to the next tab stop, tab stops are every eight columns
SHIFT/TAB	moves the cursor to the previous tab stop

Certain keyboard keys aid in the editing process by performing special functions. These keys, along with their functions, are described in Table 3-1.

NOTE

On certain extended function keyboards, a separate set of cursor control and page control function keys are provided. Usually, the SHIFT/INSERT and SHIFT/DELETE functions do NOT work with this second set of keys. This is because the status of the SHIFT key is not transmitted to the computer for these keys.

3.5 FUNCTION KEY FUNCTIONS

The function key functions provided by MPEDIT enable you to perform more sophisticated editing with only a few keystrokes. Each function key function is described in detail in this section.

3.5.1 COPY/EXTRACT

This key provides a means of "snipping" out a piece of text which can be later added to a different location. To use COPY/EXTRACT, move the current line to the first line you wish to copy or extract and press the **COPY/EXTRACT** function key. The function key labels will change so that key 1 says **COPY TO HERE**, key 4 says **EXTRACT TO HERE** and key 8 says **EXIT**. Using the down arrow, move the current line to the end of the area you wish to copy. Notice that all the lines in the selected range are highlighted. When you reach the last line of the copy area, press the key marked **COPY TO HERE** or the key labeled **EXTRACT TO HERE**. The display will return to normal. A copy of the lines within the selected range has now been placed in a holding area so that the copy may be "pasted" into the program in another location.

The copy portion of the **COPY/EXTRACT** function also works with the PREVIEW FILE function, allowing you to copy lines from another file into the one being edited.

3.5.2 PASTE

This function allows a section of text which has been "snipped" out using the COPY/EXTRACT function to be placed at any desired location in the program. Move the current line bar to the desired location and press the PASTE function key. The copied lines will be inserted into the program above the current line bar.

3.5.3 SEARCH

This function allows you to search throughout the text of your program for a particular sequence of characters. Pressing this function key causes MPEDIT to request the character string you wish to search for. Type the characters and press the **Enter** key. Searching begins at the line following the current line and continues until the desired string is found. If the requested string is found, the current line is moved to the line containing the string. If the requested string is not located, the message "Not Found" is displayed.

Subsequent search requests will cause the previous search string to the displayed under the message "Enter Search String". Simply press the **Enter** key to search for the same string again. To search for a new string, type it over the top of the old string. Be sure to blank out any excess characters from the old string.

3.5.4 REPLACE

The replace function works in a manner similar to the search function, except that the replace function allows you to replace the search string with a new sequence of characters. For example, the replace function could be used to change all occurrences of the string **"Axis_1"** to **"Axis_2"**.

Pressing this function key causes MPEDIT to first ask for the search string and then for the replace string. Searching then begins. If the search string is located, four function key options are presented:

(F4) ALL - Changes all occurrences of the search string to the replace string without intervention.

(F5) REPLACE - Changes this occurrence and begins searching for the next one.

(F6) SKIP - leaves this occurrence unchanged and begins searching for the next one.

(F8) RETURN TO EDIT - Returns to the editing mode.

3.5.5 APPEND FILE

This function will add the contents of a specified macroprogram source file to the end of the program currently being edited. Pressing this key will cause MPEDIT to ask for the name of the program file to be appended. Type the name and press the **Enter** key. The BROWSE FILES capability can also be used with APPEND.

3.5.6 FUNCTION DESCRIPTION

Pressing this function key will display the current list of Macroprogram instructions. To view the details of an individual instruction, position the highlight by using the up and down arrow keys to the desired instruction and press the **Enter** key. To exit from these descriptions, press the EXIT DESCRIPT function key.

3.5.7 QUIT NO SAVE

Pressing this function key exits from this session of editing and returns to the Macroprogram Development System selection menu without saving changes made during this edit session. The message

WARNING: FILE WILL NOT BE SAVED. ARE YOU SURE?

will be displayed. If a **Y** is typed, the original program file is reloaded from the disk, and changes are discarded. An **N** response will return control to the edit mode.

3.5.8 EXIT AND SAVE

Pressing this key also exits from this session of editing. However, all changes made during this edit session will be saved onto disk. Before rewriting the edited file to disk, the system prompts for a file name. The current name is shown as a default. To save the new program under the same file name as the original, press the **Enter** key. If you wish to assign a new file name, type the new name over the top of the old name.

When the Toolkit saves the file, it renames the original copy from "filename".PRG to "filename".BAK. In this way you have at least one backup copy.

If any changes were made to the program during the editing session, the Toolkit will automatically invoke MPCPL to compile the program.

3.5.9 GO TO LINE

Moves the highlight bar to a specific line number. This function can be used with the error printout from the MSC Macroprogram Compiler to quickly move to a specific line.

3.5.10 DISP ERRORS

When the MSC Macroprogram Compiler is used to compile a Macroprogram, it produces a list of any errors found. MPEDIT will read this list if it is present, and display the message "Errors Found" on the screen when it starts up. If you wish to review the error list, press the **DISP ERRORS** function key. This function key does not appear if no error list exists for the program being edited.

3.5.11 NEXT ERROR

This function key is used after a Macroprogram has been compiled. It functions like the search key, except that is causes MPEDIT to move to the next line containing an error. Note that lines containing errors will normally be shown as blinking on the screen (some monitors are incapable of displaying blinking characters).

This function key does not appear if no error file exists or the error file is empty for the program being edited.

3.5.12 INDENT

This function will align the instructions in a Macroprogram into formatted columns. Labels in column 1, instructions in column 16, parameters in column 31 and comments in column 61. Column formatting is **NOT** required for successful program compilation and execution.

3.5.13 PREVIEW FILE

This function provides a means of reviewing the contents of one file while editing another. A section of the reviewed file may be copied into the file being edited in a manner similar to the COPY/EXTRACT function. PREVIEW FILE can be useful to inspect other macro programs similar to the one being edited. To use the preview function, press the corresponding function key. MPEDIT will respond by asking for a file name to preview. Type the desired file name and press Return. Note that the BROWSE FILES function can be used to select a file for previewing. It is also possible to use the COPY portion of COPY/EXTRACT to copy lines from a file being previewed into the program currently being edited.

3.5.14 HELP

Pressing this key will display an abbreviated screen listing of the functions of special keyboard keys. By using the **PgDn** and **PgUp** keyboard keys, you can view further information concerning MSC flags, I/O definitions, and status words. Press the **EXIT** function key to return to the editing screen.

3.5.15 OOPS

This function allows the **last** line of text deleted to be added back. This function is only available in the IBM PC version and works only if the line was deleted using the **SHIFT/DEL** keyboard keys.

3.5.16 SAVE POSITION

This function allows you to mark a line in the file to return to should you wish to perform other functions on the remainder of the file. This function is used in conjunction with the BACK TO POSITION function. An example of the use of this feature is: You are presently at the fiftieth line of a given file and you now wish to search through the entire file for a particular character string. Following the search however, you wish return to the fiftieth line. You would press the SAVE POSITION function key, then proceed to do your search. When the search was completed, you would press the BACK TO POSITION function key.

3.5.17 BACK TO POSITION

This function is used in conjunction with the SAVE POSITION and allows you to reposition the current line indicator to the previously marked (SAVED) line. Refer to Section 3.5.16 above for an example of the use of this function key.

3.5.18 CLEAR LINE

This function allows you to erase a portion of a line or even an entire line. Position the cursor to the desired character. Now press this function key and all characters from the cursor to the right end of the line will be erased. This feature is only available in the IBM PC version.

3.5.19 CLEAR DISPLAY

This function key allows you to erase the entire edit work area. Be very careful when using this feature as there is no way to restore the work area other than using the **QUIT NO SAVE** function.

4.0 MPCPL - MACRO PROGRAM COMPILER

4.1 INTRODUCTION

The Macroprogram Compiler (MPCPL) converts MSC Macroprograms from human readable instructions to numeric codes which can be interpreted and acted upon by the MSC. The Compiler is automatically executed on exiting from MPEDIT, or prior to running the debugger if the system cannot find the machine control program file for the program you are editing.

In addition, MPCPL produces information which is used by the Macroprogram Editor (MPEDIT) and Macroprogram Debugger (MPDEBUG) to simplify the programming and testing processes. Optionally, MPCPL may also produce a detailed program listing file.

4.2 USING MPCPL

MPCPL is invoked automatically by the Toolkit whenever compilation is necessary.

4.3 MACROPROGRAM LINE FORMAT

Macroprogram lines can be made of up to four (4) parts, as shown in figure 4.1. These parts are:

Figure 4.1 - MACROPROGRAM INSTRUCTION FORMAT

label instruction *parameters* *comment*

1 .Label - A label can be from one (1) to twelve (12) characters in length, and may consists of both upper and lower case letters, numbers and the underscore character. Labels must begin in column 1 of the program line. Labels are optional. If no label is used, at least one blank must be at the beginning of the line.

2. Instruction - This part of the line consists of any valid Macroprogram Language command. Instructions must be typed in lower case letters. At least one blank must precede the instruction and at least one blank must follow the instruction.

3. Parameters - This part of the program line consists of the information, if any, processed by the instruction. It must be separated from the instruction by one or more spaces.

4. Comment - This field can contain any explanatory information about the program line. It must be separated from the preceding field by at least one blank. Program lines which contain only comments must have an exclamation point (!) in column 1.

4.4 MACRO COMPILER OUTPUT FORMAT

MPCPL produces the following outputs:

1. Listing File (Optional) - The listing consists of:

    a. An EQUATE Table - This table lists all symbols defined in **equ** statements. Its contents are symbol name, decimal symbol value, and hexadecimal symbol value.

    b. A LABEL Table - This table contains all program statement labels and their equivalent addresses.

    c. A CONSTANTS Table - This consists of a list of all constants used. This table consists only of address and values.

    d. A DATA Table - This table contains all the data variables used in the macroprogram. Each entry in this table consists of the variable name, followed by its address in both decimal and hexadecimal formats.

    e. A PROGRAM Listing - This portion consists of a listing of each program line together with the MSC numerical equivalent of the program line. Any errors detected in a program line will be listed just after the line containing the error.

2. Error Summary File - This automatically created file contains a summary of any errors detected in the compilation process. It is used by MPEDIT to assist in locating and correcting errors.

3. Symbol Table File - This optional diskette file contains all symbol, label, and data definitions. It is used by MPDEBUG to assist in the debugging process.

4. Binary Program File - This file contains the compiled program in MSC Machine Instruction Format. It can be transmitted to the MSC by MPDEBUG (or other programs), and the MSC can perform the instructions contained in it.

5. Debugger File - This file contains information used by MPDEBUG to translate trace information into source program lines.

4.5 SPECIAL MPCPL INSTRUCTIONS

MPCPL recognizes certain instructions which are not acted upon by the MSC, but simplify the process of defining data and constants. Chapter Six of this manual contains a description of these statements.

5.0 MPDEBUG - MACRO PROGRAM DEBUGGER

5.1 INTRODUCTION

The Macroprogram Debugger (MPDEBUG) provides a means of transmitting a compiled MSC Macroprogram from a personal computer to the MSC and monitoring its execution. MPDEBUG provides the ability to:

1. Inspect and modify values in the Macroprogram data area.
2. Test and modify the status of MSC flags.
3. Trace program flow and execution.
4. Test the MSC Macroprogram status word.
5 .Inspect the status of each motor axis card.
6. Review program source file, listing file, symbol table, error file.
7. Transfer the resident Macroprogram to an EPROM chip via the PROM POCKET.
8. Place each Axis Controller in the MSC System Unit into Test Mode.

MPDEBUG is able to reference information using the symbols which were used when the Macroprogram being debugged was written. This simplifies the debug process by eliminating the need to converse in decimal or hexadecimal numbers.

This chapter describes the use and features of MPDEBUG.

NOTE

The MSC controller **must** be connected to your computer before selecting MPDEBUG from the Macroprogram Development system selection menu. See IB11C001, the MSC-850 System Unit, for cable information.

5.2 USING MPDEBUG

MPDEBUG is accessed from the main Toolkit menu by highlighting the appropriate option and pressing the **Enter** key. MPDEBUG will verify that an MSC controller is connected to the serial port specified in System Configuration, and if so, will continue to the main MPDEBUG screen.

The top line of the main MPDEBUG screen, shows the name of the program currently residing in the MSC (if any), and the current status of the MSC. Initially, this status is STARTUP. The current status of the MSC and its card configuration will be displayed on the lower portion of the display. (See Section 5.4.1.6 on MACRO STATUS below).

5.3 MPDEBUG CONVENTIONS

Many MPDEBUG commands request data values, flag numbers, etc. These values may be typed in either numerically or as valid labels as defined in the Macroprogram source. For example, if the Macroprogram source defined axis 1 as "rotor", the word "rotor" could be used as a response when requesting the status of axis 1. MPDEBUG Data values may be entered as numbers or as valid arithmetic expressions. For example, to express 3.5 motor turns, where 4096 bits represents one motor turn, the expression

3.5*4096

could be entered. MPDEBUG will perform the computation and use the result.

One function of MPDEBUG is available at all times. This is **STOP PROGRAM**. The **STOP PROGRAM** function commands the MSC to stop executing the current Macroprogram and to send a **f_decel** instruction to all Controller cards.

5.4 MPDEBUG FUNCTIONS

The functions of MPDEBUG are broken into 6 sections. Each section and the functions it provides is described below. To select one of the major function areas, press its associated function key. To return to the main MPDEBUG menu, press the **F9** function key.

5.4.1 READ FUNCTIONS

The READ functions provide a means of retrieving information from the MSC and displaying it on the computer's screen.

5.4.1.1 READ DATA

This function reads data from the MSC data area. Selecting **READ DATA**, will cause MPDEBUG to ask for a data address. Enter the address symbolically or numerically. MPDEBUG will read the data from the appropriate location in the MSC and display its value in decimal and in hexadecimal. You may continue in this function by typing another address, or exit by pressing the **EXIT READ** function key.

5.4.1.2 READ DATA CONTINUOUS

This function performs just like **READ DATA** except that the specified location is read repeatedly and displayed on the screen until the **EXIT READ** function key is pressed. Up to four data locations may be selected to be read continuously.

5.4.1.3 READ FLAG

This function will retrieve the status (ON = 1, OFF = 0) of the desired flag from the MSC. The flag number may be entered as a number or as a label. See Chapter 8 for further discussion of flags.

5.4.1.4 READ FLAG CONTINUOUS

This function will continuously read and display the status of a specified MSC flag. Press the **EXIT READ** function key to stop this function. Up to four flags may be read continuously.

5.4.1.5 AXIS STATUS

This function continuously reads the status bits for the desired axis (Controller) and displays appropriate messages if any of the status bits are set. If no status bits are set, no messages will appear.

5.4.1.6 MACRO STATUS

This command reads and displays the status information from the MSC System Unit. The screen shows the name of the currently loaded Macroprogram and the date and time this Macroprogram was compiled. The lower portion of the screen displays the name and software revision levels for each Controller in the MSC System unit. A description of any bits set in the Macroprogram Status Word also appears. The MACRO STATUS function does NOT continuously update the screen.

5.4.2 WRITE FUNCTIONS

The Write functions provide a means for changing Macroprogram data locations and flags.

5.4.2.1 WRITE DATA

This function writes data into the MSC data area. MPDEBUG will ask first for a data address. Enter the desired address symbolically or numerically. MPDEBUG will then ask for a data value. Enter this value as a single number or an arithmetic expression.

5.4.2.2 WRITE DATA CONTINUOUS

This function allows continued updating of a particular MSC data location. It functions similarly to **WRITE DATA** except that after sending the specified value to the MSC, MPDEBUG will immediately request another value to be written. Exit the **WRITE CONTINUOUS** function by pressing the function key labeled **EXIT WRITE**.

5.4.2.3 WRITE FLAG

This function will set or clear the desired flag. The flag number is entered either as a symbol or a number. To change the state of the specified flag, press the appropriately labeled function key. Alternately, type an **S** to set the flag or a **C** to clear it. If any other character is typed, the status of the flag is not changed.

NOTE

Writing to an I/O flag assigns that flag as an output flag and will no longer respond to an external input.

5.4.2.4 WRITE FLAG CONTINUOUS

This function allows the user to continuously change the state of a selected flag. Exit this function by pressing **EXIT WRITE**.

5.4.3 TRACE FUNCTIONS

The MSC has the ability to "remember" the last 112 instructions that it executed. MPDEBUG can tell the MSC when to start and stop this "memory" and can read and display the values contained in it. These **TRACE** functions are described below.

The trace functions request two address values in order to know how to perform the trace. The first address requested is the focal point for the trace. (The prompt message will depend on the trace mode being used.) The second address requested is the **START TRACE AT** address. The MSC will not activate its trace memory until this address is encountered. Note that simply pressing the **Enter** key in response to the prompt address will cause the MSC to begin the trace immediately. This feature can be used as shown in the following example. Suppose you wish to trace execution of a subroutine labeled **check_status**, but only after the instruction labeled **run_machine** had executed. The proper key sequence would be:

1. Press the **TRACE AFTER** function key.

2. In response to the **TRACE AFTER ADDRESS:** prompt, enter **check_status**.

3. In response to the **START TRACE AT (PRESS ENTER TO START IMMEDIATELY)** prompt, enter **run_machine**.

This sequence of steps tells the MSC to watch for the execution of the instruction at the label **run_machine**, and, after encountering it, to start watching for the instruction labeled **check_status**. When the **check_status** address is encountered, the next 112 instructions will be saved, transmitted to MPDEBUG, and displayed on the screen.

Address values are easily entered by using the appropriate Macroprogram statement label. A list of labels is provided as part of the Macroprogram listing, and can be viewed using the **VIEW SYMBOLS** or the **VIEW SOURCE** function of MPDEBUG. Trace points between statement labels may be referenced by their numeric address. These addresses can be determined using the **VIEW SOURCE** function.

5.4.3.1 TRACE BEFORE

This function allows you to view the Macroprogram instructions that were executed before reaching the specified label.

5.4.3.2 TRACE ABOUT

This function will cause the instructions just before and just after the specified label to be traced.

5.4.3.3 TRACE AFTER

This function will trace the Macroprogram instructions executed after reaching the specified label.

5.4.3.4 TRACE CURRENT

This function commands the MSC to trace the next 112 instructions executed. It can be useful if you are not sure which part of your program is currently being executed.

5.4.3.5 STOP TRACE

Stop Trace commands the MSC to stop the currently active trace. It would most often be used when the specified trace does not finish naturally. For example, if you set up a trace after the label **set_zero**, and the program never got to the specified label, attempts to **READ TRACE** would produce only the message "Trace still running". **STOP TRACE** would then be used to tell the MSC to stop performing the trace. When the **STOP TRACE** function is used, the trace buffer may not contain meaningful information.

5.4.3.6 READ TRACE

This function reads and displays the results of the last trace executed by the MSC. The trace screen may be scrolled up and down by using the cursor keys, the **PgDn** and **PgUp** keys, and the **Home** key. If the trace last specified has not completed, the message "Trace still running" will be displayed.

5.4.4 MSC COMMANDS

This group of commands deals with transmitting Macroprograms to the MSC, starting program execution, and setting the MSC into various modes of operation.

5.4.4.1 STOP PROGRAM

This function causes the currently executing Macroprogram to be stopped. An **f_decel** command is sent to each Controller card. NOTE: The **STOP PROGRAM** function key is available on all MPDEBUG menus.

5.4.4.2 RESET

This function sends a serial RESET command to the MSC. The RESET will clear all flags, reset all Controller cards, and erase any Macroprogram and data resident in the MSC. This function must be preceded by a **STOP PROGRAM** function.

5.4.4.3 SEND PROGRAM

This function will transmit the currently loaded Macroprogram to the MSC. A **RESET** function must be performed prior to using the **SEND PROGRAM** function.

5.4.4.4 START PROGRAM

This function will begin execution of the Macroprogram currently resident in the MSC. This function will not execute if no program has been sent to the MSC.

5.4.4.5 PROM OPTIONS

This function provides access to a sub-menu used to copy the Macroprogram currently resident in the MSC to an EPROM. NOTE: It is possible for the program resident in the MSC to be different from the program resident in the Toolkit.

To copy the resident program to EPROM, perform the following steps:

1. If the resident program is running, press **F1**, **STOP PROGRAM**.

2. The PROGRAM MODE switch on the MSC PROM Pocket must be set to the **Program** position(the MSC-850/32 controller does not have a PROGRAM MODE switch).

3. Insert the EPROM to be programmed in the PROM Pocket. The EPROM must be an INTEL D27256-1 UV erasable EPROM, or equivalent.

4. Press function key **F2**, **BURN PROM**. The message

BURNING PROM.....

will appear at the top of the screen. Depending on the size of the program being copied to EPROM, the message may remain on the display for up to 2 or 3 minutes.

While the EPROM is being programmed, the green LED on the PROM Pocket will be illuminated. When using the MSC-250 or MSC-850/32 controllers, the status display will indicate that the EPROM is being programmed.

On successful completion of the programming operation, the message

PROM Operation Successful

will appear. If an error is detected during programming, refer to Table 15.1 for an explanation of the error code.

5.4.4.6 TEST MODE

This function places each axis in the MSC System Unit into the test mode. Refer to the MSC System Manual Test Procedure section.

5.4.4.7 SET AUTOSTART

This function sets the **AUTOSTART** bit in the MSC status word and, if necessary, begins execution of the Macroprogram residing in MSC non-volatile memory. The MSC operating system firmware tests the **AUTOSTART** bit on power-up. If it is turned on, the Macroprogram currently stored in non-volatile memory will be executed.

5.4.5 VIEW FUNCTIONS

The view functions provide a means of displaying various pertinent files on the screen during the editing process. You may move from place to place in the file being displayed using the **PgDn** and **PgUp** keys.

5.4.5.1 SOURCE

This function displays the source program file (created by MPEDIT) for the program currently being debugged. Program addresses are displayed for each instruction in the program. This is a convenient way to determine an address value to use with trace functions.

5.4.5.2 EQUATE TABLE

This function displays all constants defined using the **equ** compiler directive.

5.4.5.3 LABEL TABLE

This function displays all program labels and their corresponding addresses.

5.4.5.4 CONSTANTS

This function displays the constants defined in the Macroprogram along with their data addresses.

5.4.5.5 DATA TABLE

This function displays all data locations defined using compiler directives such as **begin_data**, **dim**, or **integer**. The format for this display shows the variable name, its decimal address, and its hexadecimal address.

5.4.5.6 LAST TRACE

This function displays the information from the last trace executed by the MSC. If no tracing has been done, an appropriate message is displayed.

5.4.6 BLOCK FUNCTIONS

Block commands are used to read and write data areas. They are helpful when a large amount of continuous data has to be modified or viewed.

When the **BLOCK** function key is pressed, the system responds by showing three additional function keys; **STOP PROGRAM**, **READ DATA**, and **WRITE DATA**.

5.4.6.1 READ DATA

This function key allows you to select the beginning of a data area to be viewed. The data address is entered and the system will display the current contents of each of a series of data locations. By pressing the **PgDn** key, you can view the next group of continuous data locations. To exit BLOCK READ, press the EXIT BLK READ function key.

5.4.6.2 WRITE DATA

This function key allows you to select the beginning of a data area to be viewed and, optionally, modified. The desired data address is entered and the system displays the Macroprogram data area beginning with the selected data address. To modify a data value, use the cursor keys to move the highlight to the desired variable. Type the new value and press the enter key. To change between decimal and hexadecimal format, press the appropriate function key. To exit BLOCK WRITE, press the EXIT BLK WRITE function key.

6.0 INTRODUCTION TO MACROPROGRAMMING LANGUAGE

6.1 BASIC CONCEPTS

A Macroprogram is an organized group of instructions that can be executed by an MSC system unit. Macroprograms reside in a non-volatile memory in the system unit.

In the MSC-850/32, there are 64,000 bytes allocated for the program area and an additional 64,000 bytes allocated for the data area.

In the MSC-250, 32,000 bytes of storage are available for the combined program and data areas.

In an MSC-800 and MSC-850, 16,000 bytes of storage are available for the combined program and data areas. This memory space is dynamically allocated as the program is compiled. The size of each area may vary, so long as the sum of the program and data areas do not exceed 16,000 bytes.

6.2 INSTRUCTION FORMAT

Macroprogram instructions are, in many ways, similar to the instructions of the BASIC computer language. Each instruction consists of a statement label, an operation, a list of parameters, and an optional comment. This basic format is illustrated in Figure 6.1.

Figure 6.1 - MACROPROGRAM INSTRUCTION FORMAT

label instruction *parameter list* *comment*

The label portion of the instruction is not always required, depending on the type of instruction used. It is required for instructions which define variables, text strings, arrays, or equated values.

The parameter list part of an instruction line is also not always required. However, there are usually one or more parameters for an instruction. In the example in Figure 6.2, two parameters are required.

Figure 6.2 - INSTRUCTION WITH TWO PARAMETERS

label set_speed axis_1,300 *comment*

6.3 COMMENTS

Comments provide a means for a programmer to describe the functions being performed by a particular instruction or group of instructions. It is important to note that comments do not use any of the storage within the Macroprogram memory space. The liberal use of comments is highly recommended. There are two ways to implement comments within a Macroprogram:

1. An entire line of text can be dedicated as a comment line by placing an exclamation point in the first character position of the line.

2. Comments can be placed at the end of an instruction line by leaving at least one space between the last parameter in the instruction and the beginning of the comment.

Comments of both formats are included in the sample Macroprogram shown in Figure 6.3.

6.4 BLANK LINES

Blank lines may be entered in the Macroprogram to make the program more readable. For example, a subroutine might be separated from the main body of the Macroprogram by one or more blank lines. As with comments, blank lines do not use any storage within the Macroprogram.

6.5 LABEL LINES

Program lines containing only a label are allowed. It is often handy to place a line containing only a label just ahead of an instruction it references. In this way, it is easier to insert new instructions in the program if necessary during the testing process. Lines containing only a label use no storage within the Macroprogram.

Figure 6.3 - SAMPLE MACROPROGRAM

```
LABEL       OPERATION     PARAMETERS          COMMENT

            msc_type      850
            declare       ON

pos_1       equ           81920               two turns CW
pos_2       equ           -81920              two turns CCW
rack        equ           1                   use axis 1
rack_down   equ           93                  down flag
rack_busy   equ           94                  busy flag
fault_lite  equ           0                   I/O zero

! do initialization functions

            turn_off      fault_lite          show no fault

            drive_on      rack                enable drive
            set_speed     rack,50             500 rpm speed
            set_ac_dc     rack,20             rev/sec/sec

! set a local zero at current position

            set_local     rack

! begin main program loop

restart
            position      rack,pos_1          go 2 turns CW
loop1
            if_stat_on    rack_down,fault     check error
            if_stat_on    rack_busy,loop1     wait for done

            position      rack,pos_2          go 2 turns CCW

loop2
            if_stat_on    rack_down,fault     check error
            if_stat_on    rack_busy,loop2     wait for done

            goto          restart             do it all again

fault
            turn_on       fault_lite          signal error
            sys_return
```

7.0 COMPILER DIRECTIVES

7.1 DESCRIPTION

Compiler directives are a class of instructions which simplify such operations as defining constants, data arrays and cams, or which cause the MPCPL program to function in a particular way. These instructions have no direct effect on the MSC controller. Their purpose is to simplify the programmer's task.

A list of compiler directive instructions and their formats is shown in Table 7.1.

Table 7.1 - COMPILER DIRECTIVES

Instruction	Format			Ref. Page
begin_cam	label	begin_cam		116
begin_data	label	begin_data		117
cam		cam	value,value,etc.	122
data		data	value,value,etc.	134
declare		declare	mode	135
dim	label	dim	controller#,size	138
end_cam		end_cam		146
end_data		end_data		147
equ	label	equ	value	149
integer	label	integer		203
msc_type		msc_type	system_type	217
text	label	text	"ASCII string"	291

8.0 FLAGS

8.1 DESCRIPTION

A flag is a bit in memory representing the current state of a timer, axis status, I/O or other condition. The MSC has reserved a storage area in memory for 256 flags. A flag can be either on (1) or off (0).

There are a number of Macroprogram instructions dealing with flags. In fact, taken as a group, flag instructions comprise the largest subset of macroprogram instructions.

There are flag instructions to read inputs, to turn outputs on and off, to start and stop timers, to read motor activity and fault conditions and to read, set and clear user program switches.

Tables 8.1 through 8.9 list the flags available in the MSC family of controllers.

NOTE:

The following are special considerations when using the MSC-250:

1) The MSC-250 controller can use only two (2) I/O Expander units and therefore does not use I/O flags 48 - 63.

   Flags 64 - 71 are used as software PLS flags.

2) The motor status flags 128 - 175 are reserved for extended status indicators for axes 1 - 3 respectively on the MSC-250.

3) The motor status flags 176 - 207 are not used in the MSC-250. These **cannot** be used as additional user flags.

FLAG #	FUNCTION	CATEGORY
0 - 7	ON BOARD I/O, PERMANENTLY ASSIGNED	INPUT/ OUTPUT
8 - 23	I/O EXPANDER ADDRESS CODE "A"	
24 - 39	I/O EXPANDER ADDRESS CODE "B"	
40 - 55	I/O EXPANDER ADDRESS CODE "C"	
56 - 71	I/O EXPANDER ADDRESS CODE "D"	
72 - 79	PROGRAMMABLE TIMERS	TIMER
80 - 95	CONTROLLER #1 STATUS FLAGS	CONTROLLER STATUS
96 - 111	CONTROLLER #2 STATUS FLAGS	
112 - 127	CONTROLLER #3 STATUS FLAGS	
128 - 143	CONTROLLER #4 STATUS FLAGS	
144 - 159	CONTROLLER #5 STATUS FLAGS	
160 - 175	CONTROLLER #6 STATUS FLAGS	
176 - 191	CONTROLLER #7 STATUS FLAGS	
192 - 207	CONTROLLER #8 STATUS FLAGS	
208 - 255	GENERAL PURPOSE USER FLAGS	USER

Table 8.1 - MSC-800, MSC-850, MSC-850/32 INTERNAL STATUS FLAGS

FLAG #	FUNCTION	CATEGORY
0 - 15	ON BOARD I/O, PERMANENTLY ASSIGNED	INPUT/ OUTPUT
16 - 31	I/O EXPANDER ADDRESS CODE "A"	
32 - 47	I/O EXPANDER ADDRESS CODE "B"	
48 - 63	NOT USED	
64 - 71	SOFTWARE PLS FLAGS	
72 - 79	PROGRAMMABLE TIMERS	TIMER
80 - 95	CONTROLLER #1 STATUS FLAGS	CONTROLLER STATUS
96 - 111	CONTROLLER #2 STATUS FLAGS	
112 - 127	CONTROLLER #3 STATUS FLAGS	
128 - 143	NOT USED	
144 - 159	NOT USED	
160 - 175	NOT USED	
176 - 191	NOT USED	
192 - 207	NOT USED	
208 - 255	GENERAL PURPOSE USER FLAGS	USER

Table 8.2 - MSC-250 INTERNAL STATUS FLAGS

#1	#2	#3	#4	#5	#6	#7	#8	MEANING
80	96	112	128	144	160	176	192	MDU FAIL
81	97	113	129	145	161	177	193	LOCK PENDING
82	98	114	130	146	162	178	194	FOLLOWING ERROR
83	99	115	131	147	163	179	195	MASTER/TEST MODE
84	100	116	132	148	164	180	196	CAM/PW PROFILE SIZE EXCEEDED
85	101	117	133	149	165	181	197	COMMAND INVALID IN THIS STATE
86	102	118	134	150	166	182	198	LOSS OF RESOLVER
87	103	119	135	151	167	183	199	AXIS TIME-OUT
88	104	120	136	152	168	184	200	MOTOR JOGGING
89	105	121	137	153	169	185	201	MOTOR INDEXING
90	106	122	138	154	170	186	202	CALCULATING
91	107	123	139	155	171	187	203	HWI ARMED
92	108	124	140	156	172	188	204	FORCE DECEL IN PROGRESS
93	109	125	141	157	173	189	205	AXIS DOWN
94	110	126	142	158	174	190	206	AXIS BUSY
95	111	127	143	159	175	191	207	MASTER/SLAVE LOCK

Table 8.3 - ACR-850 CONTROLLER STATUS FLAGS

#1	#2	#3	#4	#5	#6	#7	#8	MEANING
80	96	112	128	144	160	176	192	MDU FAIL
81	97	113	129	145	161	177	193	LOCK PENDING
82	98	114	130	146	162	178	194	FOLLOWING ERROR
83	99	115	131	147	163	179	195	MASTER/TEST MODE
84	100	116	132	148	164	180	196	CAM/PW PROFILE SIZE EXCEEDED
85	101	117	133	149	165	181	197	COMMAND INVALID IN THIS STATE
86	102	118	134	150	166	182	198	MARKER OUT OF LIMIT
87	103	119	135	151	167	183	199	AXIS TIME-OUT
88	104	120	136	152	168	184	200	MOTOR JOGGING
89	105	121	137	153	169	185	201	MOTOR INDEXING
90	106	122	138	154	170	186	202	CALCULATING
91	107	123	139	155	171	187	203	HWI ARMED
92	108	124	140	156	172	188	204	FORCE DECEL IN PROGRESS
93	109	125	141	157	173	189	205	AXIS DOWN
94	110	126	142	158	174	190	206	AXIS BUSY
95	111	127	143	159	175	191	207	MASTER/SLAVE LOCK

Table 8.4 - ACE-850 CONTROLLER STATUS FLAGS

#1	#2	#3	#4	#5	#6	#7	#8	MEANING
80	96	112	128	144	160	176	192	MDU FAIL
81	97	113	129	145	161	177	193	PLS FLAG 16
82	98	114	130	146	162	178	194	NOT USED
83	99	115	131	147	163	179	195	PLS FLAG 17
84	100	116	132	148	164	180	196	PLS FLAG 19
85	101	117	133	149	165	181	197	COMMAND INVALID IN THIS STATE
86	102	118	134	150	166	182	198	PLS FLAG 18
87	103	119	135	151	167	183	199	AXIS TIME-OUT
88	104	120	136	152	168	184	200	MOTOR JOGGING
89	105	121	137	153	169	185	201	MOTOR INDEXING
90	106	122	138	154	170	186	202	CALCULATING
91	107	123	139	155	171	187	203	NOT USED
92	108	124	140	156	172	188	204	FORCE DECEL IN PROGRESS
93	109	125	141	157	173	189	205	AXIS DOWN
94	110	126	142	158	174	190	206	AXIS BUSY
95	111	127	143	159	175	191	207	MASTER/SLAVE LOCK

Table 8.5 - MCF-850 CONTROLLER STATUS FLAGS

#1	#2	#3	#4	#5	#6	#7	#8	MEANING
80	96	112	128	144	160	176	192	MDU FAIL
81	97	113	129	145	161	177	193	PLS FLAG 16
82	98	114	130	146	162	178	194	NOT USED
83	99	115	131	147	163	179	195	PLS FLAG 17
84	100	116	132	148	164	180	196	PLS FLAG 19
85	101	117	133	149	165	181	197	COMMAND INVALID IN THIS STATE
86	102	118	134	150	166	182	198	PLS FLAG 18
87	103	119	135	151	167	183	199	AXIS TIME-OUT
88	104	120	136	152	168	184	200	NOT USED
89	105	121	137	153	169	185	201	NOT USED
90	106	122	138	154	170	186	202	CALCULATING PLS DATA
91	107	123	139	155	171	187	203	NOT USED
92	108	124	140	156	172	188	204	NOT USED
93	109	125	141	157	173	189	205	AXIS DOWN
94	110	126	142	158	174	190	206	NOT USED
95	111	127	143	159	175	191	207	NOT USED

Table 8.6 - HPL-850 CONTROLLER STATUS FLAGS

#1	#2	#3	#4	#5	#6	#7	#8	MEANING
80	96	112	128	144	160	176	192	NOT USED
81	97	113	129	145	161	177	193	NOT USED
82	98	114	130	146	162	178	194	NOT USED
83	99	115	131	147	163	179	195	MASTER TEST MODE
84	100	116	132	148	164	180	196	NOT USED
85	101	117	133	149	165	181	197	COMMAND INVALID IN THIS STATE
86	102	118	134	150	166	182	198	NOT USED
87	103	119	135	151	167	183	199	AXIS TIME-OUT
88	104	120	136	152	168	184	200	NOT USED
89	105	121	137	153	169	185	201	NOT USED
90	106	122	138	154	170	186	202	NOT USED
91	107	123	139	155	171	187	203	NOT USED
92	108	124	140	156	172	188	204	NOT USED
93	109	125	141	157	173	189	205	AXIS DOWN
94	110	126	142	158	174	190	206	NOT USED
95	111	127	143	159	175	191	207	NOT USED

Table 8.7 - ACM-850 CONTROLLER STATUS FLAGS

#1	#2	MEANING
80	96	NOT USED
81	97	LOCK PENDING
82	98	FOLLOWING ERROR
83	99	MASTER/TEST MODE
84	100	CAM/PW PROFILE
85	101	COMMAND INVALID IN THIS STATE
86	102	NOT USED
87	103	NOT USED
88	104	MOTOR JOGGING
89	105	MOTOR INDEXING
90	106	CALCULATING
91	107	HWI ARMED
92	108	FORCE DECEL IN PROGRESS
93	109	AXIS DOWN
94	110	AXIS BUSY
95	111	MASTER/SLAVE LOCK

Table 8.8 - MSC-250 CONTROLLER STATUS FLAGS AXES 1 & 2

#3	MEANING
112	NOT USED
113	NOT USED
114	NOT USED
115	NOT USED
116	CAM/PW PROFILE SIZE EXCEEDED
117	COMMAND INVALID IN THIS STATE
118	NOT USED
119	NOT USED
120	MOTOR JOGGING
121	MOTOR INDEXING
122	CALCULATING
123	NOT USED
124	FORCE DECEL IN PROGRESS
125	NOT USED
126	AXIS BUSY
127	NOT USED

Table 8.9 - MSC-250 PSEUDO CONTROLLER STATUS FLAGS

DESCRIPTION OF CONTROLLER STATUS FLAGS

FLAG	DESCRIPTION
MDU FAIL	A test of the MDU (Multiply/Divide Unit) is made on system power up. The purpose of the test is to exercise the arithmetic functions of the MDU (shift, rotate, add, subtract etc.). This flag will be set if an error occurs while testing the unit. This flag is not used in the MSC-250 as there is no Multiply/Divide chip.
POSITION LOOP FAIL	A test of the Resolver to Digital Converter is made on system power up. This flag will be set if an error is encountered with the unit. This flag is not used in the MSC-250.
FOLLOWING ERROR	A following error occurs when the difference between the actual transducer angle and the expected transducer angle is outside the allowable range. The angles are compared every 10 ms. The allowable error is +/- 17 degrees if the motor shaft is motionless and +/- 180 degrees if the motor shaft is moving. **NOTE:** If digital compensation is being used, the allowable error while the motor shaft is moving becomes +/- 180*16/PGAIN degrees, where PGAIN is the proportional gain setting being used.
MASTER/TEST MODE	This flag will be set if the axis has been put into the Master/Test mode. All motor fault conditions will have been reset. This is the default, power on state for an axis controller. For the ACR-850, ACE-850 and MSC-250 axis controllers, the servo amplifier will have been disabled and no checks for servo following errors will be made. The analog position output will be set to be proportional to the transducer position as shown in Figure 8.1.

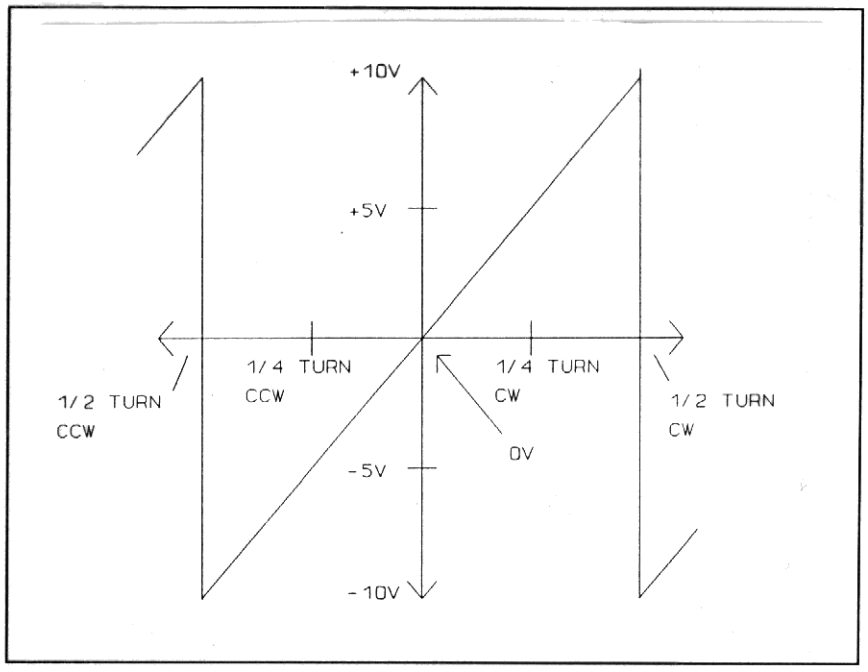

Figure 8.1 - POSITION OUTPUT vs. TRANSDUCER ANGLE

CAM SIZE EXCEEDED/ PIECEWISE PROFILE ARRAY FULL	This flag will be set if the number of cam array data elements exceeds 28K bytes, or if the number of piecewise profile segments exceeds 99.
COMMAND INVALID IN THIS STATE	This flag will be set whenever the axis is requested to execute a command that it is not capable of at that moment. This typically occurs when program steps are out of logical order. An example would be the situation where the axis is currently jogging and is then commanded to index without first completing a **f_decel** instruction and waiting for the motor to stop.
MARKER OUT OF LIMIT	This flag, specific to the MSC-850/ACE-850 and MSC-250 axis controllers, indicates that the controller observed a marker pulse from the encoder that was more than +/- 5 bits from the expected position, and that the marker correction function has been suspended. The flag will be cleared if the marker pulse is subsequently observed within the +/- 5 bit window.
LOSS OF RESOLVER	This flag, specific to the ACR-850C (SFO-8060) axis controller card, indicates that the resolver convertor has lost one or more of the required signals for conversion or the convertor has lost its ability to track the angle do to speeds in excess of 7200 RPM.
AXIS TIMEOUT	On system power up each controller card slot in an MSC controller is checked for the existence of a functional Controller card. This flag will be set for each axis that does not have a functional card installed or for those slots experiencing communications problems. This flag is not used in the MSC-250.
MOTOR JOGGING	This flag will be set when a **jog_cw**, **jog_ccw**, **track_spd**, **l_track_spd**, **vel_cw** or **vel_ccw** command is executed. This flag will be cleared after completing a **f_decel** operation.
MOTOR INDEXING	This flag will be set when a index or position command is executed. This flag will be cleared after completing the selected move.
CALCULATING PIECEWISE PROFILE	This flag will be set while a Piecewise profile is being calculated. It will be cleared otherwise.
CALCULATING PLS DATA	This flag will be set while Programmable Limit Switch data is being calculated.
HWI ARMED	The controller has received a hardware interrupt instruction and is monitoring its respective hardware interrupt input line.
FORCED DECEL IN PROGRESS	This flag will be set when an **f_decel** or **unlock** command is executed and the axis is still in motion.
AXIS DOWN	This flag will be set if a following error or axis timeout is encountered (following errors and axis timeouts are described above).

AXIS BUSY — This flag will be set when the axis is busy performing a motion related operation such as **jog_cw**, **jog_ccw**, **index**, **position**, **vel_cw**, **vel_ccw**, **track_spd**, **l_track_spd**, or **exec_profile**. If an axis card experiences an **AXIS DOWN** condition while the **AXIS BUSY** flag is set, **AXIS BUSY** will remain set.

MASTER/SLAVE LOCK — This flag will be set when a **lock** command is executed. It will be cleared when a **f_decel**, **unlock**, or **enable** command is executed.

BUSY PROCESSING — This flag is set when the axis processor is executing a lengthy command to prevent the main processor from sending another Macroprogram instruction for the specified axis controller. The following Macroprogram Instructions cause this flag to be set: **calc_cam_sum, calc_unit_cam, drive_on, drive_off, find_mrk_ccw, find_mrk_cw, over_draw, prep_profile,** and **test_mode.**

8.2 TIMERS

The MSC controllers provide 8 user programmable timers. These timers have a resolution of 10 milliseconds per "tick". Timers are used by first setting the timer to the desired number of counts or ticks. This causes the flag associated with the timer to turn on and to remain on until the specified time has passed.

User timers are decremented on every other tick of the system's 5 millisecond clock. As a result of this, these timers should be considered accurate to +/- 10 milliseconds.

8.3 FLAG INSTRUCTIONS

Instructions for operating on and testing flags are summarized in Table 8.10.

Table 8.10 - FLAG INSTRUCTIONS

Instruction	Format			Ref. Page
blk_io_in	*label*	blk_io_in	input flag#,variable	118
blk_io_out	*label*	blk_io_out	output flag#,variable	119
clr_flag	*label*	clr_flag	user flag#	129
get_status	*label*	get_status	controller#	179
if_flag_off	*label*	if_flag_off	user_flag#,address_label	190
if_flag_on	*label*	if_flag_on	user_flag#,address_label	191
if_io_off	*label*	if_io_off	I/O flag#,address_label	192
if_io_on	*label*	if_io_on	I/O flag#,address_label	193
if_stat_off	*label*	if_stat_off	status_flag#,address_label	195
if_stat_on	*label*	if_stat_on	status_flag#,address_label	196
if_tmr_off	*label*	if_tmr_off	timer_flag#,address_label	197
if_tmr_on	*label*	if_tmr_on	timer_flag#,address_label	198
set_flag	*label*	set_flag	user_flag	244
set_tmr	*label*	set_tmr	timer_flag#,ticks	278
turn_off	*label*	turn_off	I/O flag#	294
turn_on	*label*	turn_on	I/O flag#	295