ALLEN-BRADLEY
A ROCKWELL INTERNATIONAL COMPANY

Handheld Pendant Operator's Manual

IMC 120 Motion Control System

Important User Information

If you find errors, discrepancies, or inadequacies in this document, please contact:

Allen-Bradley Company, Inc.
Industrial Computer and Communication Group
Commercial Services
747 Alpha Drive
Highland Heights, OH 44143

Chapter Title Page

4 Monitoring IMC 120 Status

5 Jogging from the Handheld Pendant

List of Figures

List of Tables

1.0 Chapter Overview

This chapter introduces you to this manual and gives you the following information:

- who we wrote this manual for
- what this manual contains
- how to use this manual effectively
- where to find more information

1.1 Manual Purpose and Audience

We wrote this manual for those who must set up applications, and run and monitor them with the IMC 120 motion control system.

We assume that if you are using this manual, you know or are familiar with:

- servo control basics
- your equipment
- programming logic controllers
- IMC 120 Motion Control Systems Programming Manual

1.2 Manual Contents

This manual is designed to help you run, monitor and store application programs. Where possible, it follows a "step-by-step" approach. and gives you the information you need when you need it.

Figure 1.1 shows the "task flow" of this manual, and Table 1.A gives a brief overview of each chapter.

Figure 1.1
Task Flow Of This Manual

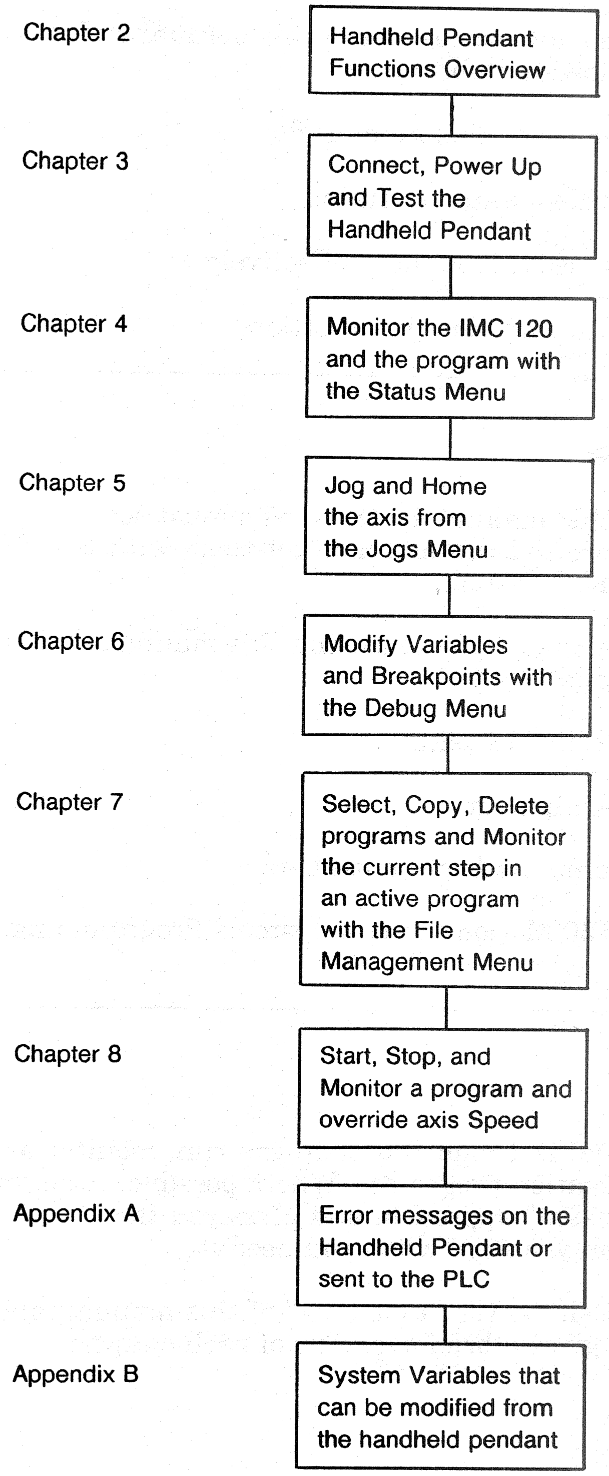

Chapter 2
Handheld Pendant Functions Overview
Chapter 3
Connect, Power Up and Test the Handheld Pendant
Chapter 4
Monitor the IMC 120 and the program with the Status Menu
Chapter 5
Jog and Home the axis from the Jogs Menu
Chapter 6
Modify Variables and Breakpoints with the Debug Menu
Chapter 7
Select, Copy, Delete programs and Monitor the current step in an active program with the File Management Menu
Chapter 8
Start, Stop, and Monitor a program and override axis Speed
Appendix A
Error messages on the Handheld Pendant or sent to the PLC
Appendix B
System Variables that can be modified from the handheld pendant

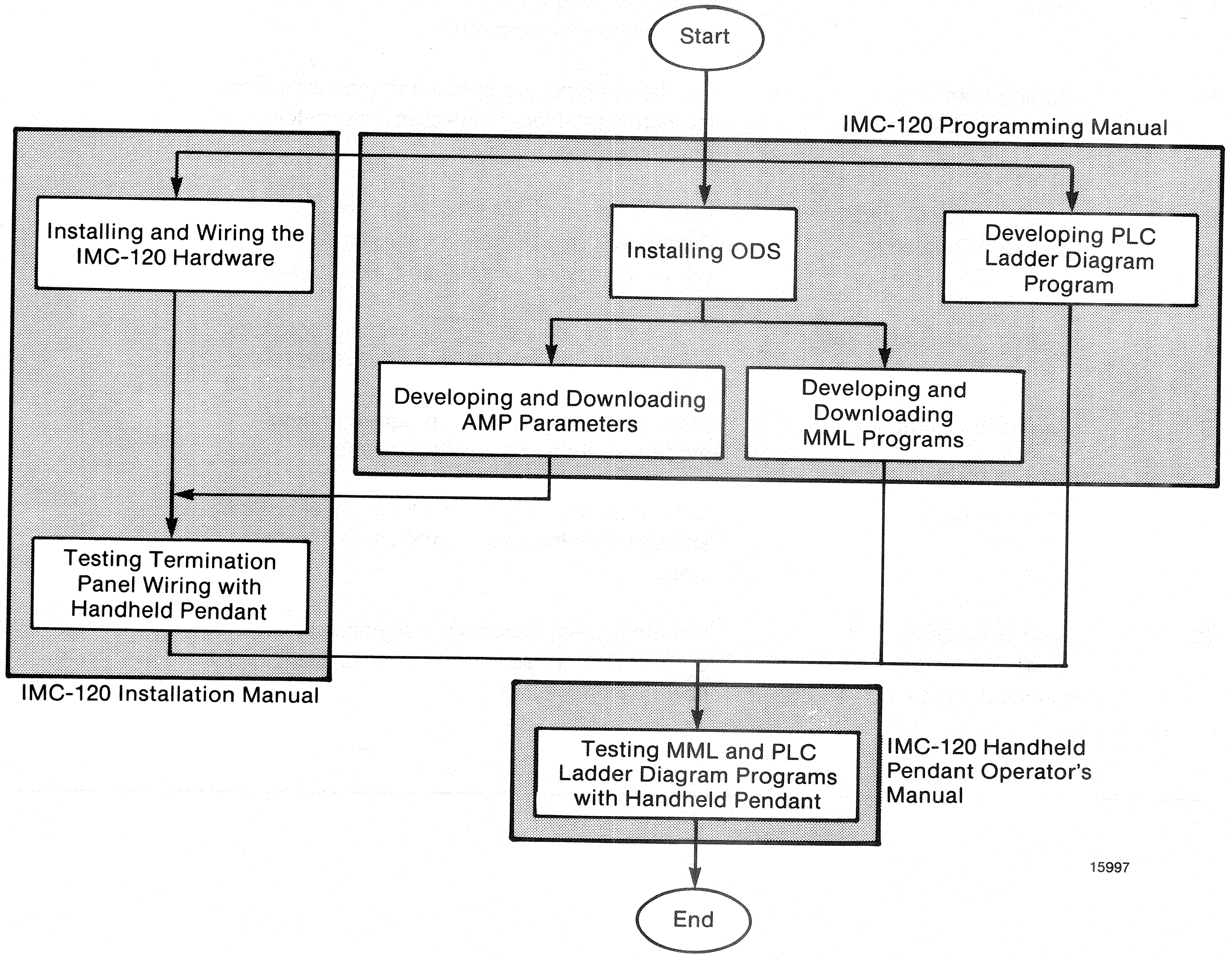
Table 1.A
What This Manual Contains

Chapter	*Title*	*Purpose*
2	Handheld Pendant Operations	Functions of the handheld pendant keyboard, switches and buttons, and menu displays
3	Power Up and Testing The Handheld Pendant	Connect the handheld pendant, power up the IMC 120 and handheld pendant, and test the handheld pendant.
4	Monitoring IMC 120 Status	Monitor the status of machine position and mode, the program that is running, the jog type and any system errors.
5	Jogging from the Handheld Pendant	Set the jog type, jog speed and jog increment for incremental jog, homing and return-to-position moves.
6	Debug Motion Management Language Programs	Initialize variables with the teach position menu. Modify variables and breakpoints with the debug menu. Dry run an MML program
7	Program File Management	Select, copy, delete and rename a program or monitor the current step in an active program
8	Automatic Operation	Start, stop, and monitor a program in auto mode and set and override axis speed
A	Error Messages	List the error messages in alphabetical order and describe the error condition and recovery steps
B	System Variables changed from the handheld pendant	List the system variables in alphabetical order and their use, range, data type, and how to access them

1.3 Using This Manual

This manual is one of a series of manuals designed to help you install, program, test and operate an IMC 120 motion control system. Figure 1.2 show you how this manual fits into the way we have provided this information.

Figure 1.2
Where This Manual Fits In


Start
IMC-120 Programming Manual
Installing and Wiring the IMC-120 Hardware
Installing ODS
Developing PLC Ladder Diagram Program
Developing and Downloading AMP Parameters
Developing and Downloading MML Programs
Testing Termination Panel Wiring with Handheld Pendant
IMC-120 Installation Manual
Testing MML and PLC Ladder Diagram Programs with Handheld Pendant
IMC-120 Handheld Pendant Operator's Manual
15997
End

1.3.1 WARNINGs, CAUTIONs, and Important Information

We use the labels **WARNING, CAUTION,** and **Important** to identify the following kinds of information:

- **WARNING:** identifies information about practices or circumstances that can lead to personal injury as well as damage to the control, your machine, or other equipment
- **CAUTION:** identifies information about practices or circumstances that can lead to damage to the control, machine, or other equipment
- **Important:** identifies information that is especially important for successful application of the controller

1.3.2 Terms and Conventions

We use the following terms and conventions in this manual.

- AMP - Adjustable Machine Parameters - system parameters that specify axis and controller characteristics
- Axis - The servo driven axis is referred to as the axis. The axis may be a rotary axis or a linear axis.
- IMC 120 - The IMC 120 servo controller module (cat. no. 1771-HS) is referred to as the IMC 120.
- <KEYS> - The angle brackets denote a key on the handheld pendant keyboard. The words or symbols in the brackets are those you will see on the key.
- MML - Motion Management Language for programming the motion of the axis the IMC 120 servo controller controls
- ODS - Offline Development System - application software that lets you use certain personal computers to create AMP and MML files and download them to the IMC 120 servo controller

- PLC - The host programmable logic controller for the IMC 120.
- SDE - Syntax Directed Editor - an ODS utility for creating and editing MML programs

1.4 Finding More Information

For more information on the IMC 120 motion control system, please contact your local Allen-Bradley sales office or distributor, or refer to these related publications:

Catalog Number	Title	Publication Number
1771-HS	IMC 120 Motion Control System Installation Manual	1771-6.5.45
1771-PS7	120/220V AC Power Supply w/User Power Product Data	1771-2.123
1771-HM	IMC 120 Plug In Memory Product Data	1771-2.124
1771-HR	IMC 120 Resolver Excitation Module Product Data	1771-2.125
1771-HT	IMC 120 Termination Panel Product Data	1771-2.126
1771-HD	IMC 120 Handheld Pendant Operators Manual	1771-6.5.50
8100-HSKAR	IMC 120 Motion Control System Programming Manual	1771-6.5.51
1771-HCDOC	IMC 120 Motion Control System Installation Manual IMC 120 Handheld Pendant Operator's Manual IMC 120 Motion Control System Programming Manual	1771-6.5.45 1771-6.5.50 1771-6.5.51

2.0 Chapter Overview

This chapter introduces the handheld pendant and briefly describes how to enable and monitor handheld pendant functions. See chapters 3 through 8 for more information on these functions.

The handheld pendant does not provide any programming capability. It can only modify system variable values and system parameter values. Input and modify MML programs through the Offline Development System (ODS).

2.1 Operator Functions from the Handheld Pendant

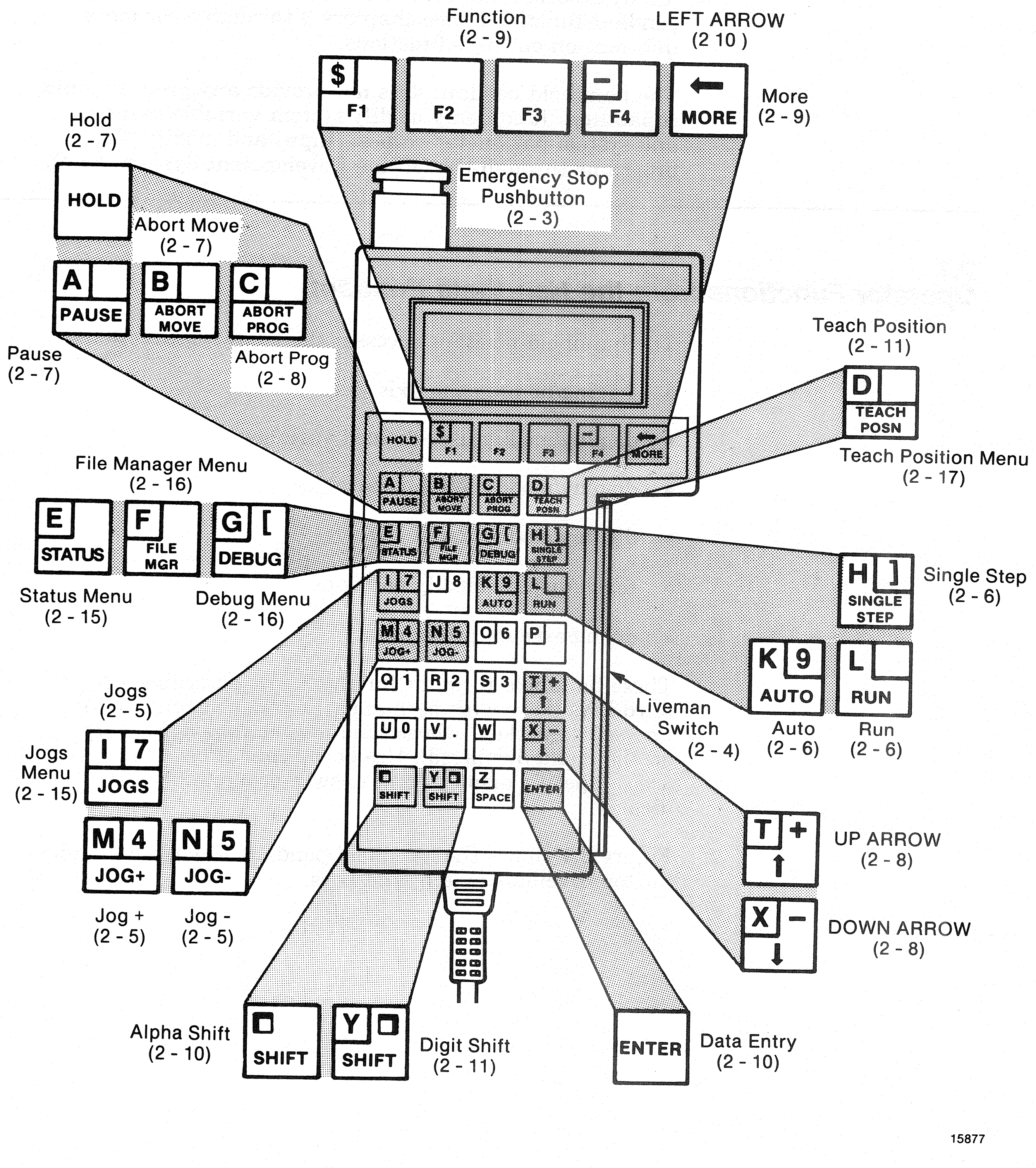
The handheld pendant accesses the following functions:

- jogging/homing the axis
- display/modify user-defined variables
- display/modify system variables
- display/modify FAST I/O
- teach/display/modify positions
- debug/execute programs

These handheld pendant functions are accessible via hard keys (keys with permanently assigned functions) and soft keys (keys whose functions change depending on the menu displayed). Use both types of keys to access specific displays and answer prompts from the display.

Figure 2.1 shows the handheld pendant Emergency Stop button, liveman switch, and keys.

Figure 2.1
IMC 120 Handheld Pendant Buttons, Switches, and Keys



2.2 Using the Emergency Stop Pushbutton and Emergency Stop Reset

Emergency Stop

When properly integrated by the system installer, the Emergency Stop button stops axis motion in case of an emergency to guard against injury to personnel and damage to equipment.

Press the Emergency Stop button on the top of the handheld pendant to stop and hold the current axis motion. When you press Emergency Stop, the servo command voltage drops to zero. The IMC 120 does not send a deceleration ramp to the servo.

If you have regenerative braking on your system, the drive motor regenerates to a stop, the axis does not coast to a stop. Check with your system installer to see if there is regenerative braking on your system.

WARNING: To guard against personal injury or damage to equipment note that the axis motion and the program can run independent of each other. Condition handlers in the MML program which can control I/O signals, continue to run after axis motion stops unless the MML program reaches end-of-program.

Important: You cannot reset Emergency Stop if you haven't downloaded system parameters (AMP) from the ODS.

Emergency Stop Reset

To reset the Emergency Stop condition:

- Correct the cause of the Emergency Stop.
- Pull out the Emergency Stop button.
- Press the user supplied Emergency Stop reset push button until the Emergency Stop relay energizes.

Press the <RUN> key on the handheld pendant to continue the program. The program resumes from the point where it stopped.

2.3 Liveman Switch

We put the liveman switch on the handheld pendant to be sure you use two hands to jog the axis or to input data. Data transmission from the pendant stops if you release the liveman switch. A jog move also stops because it depends upon this data transmission.

Jog moves work only when you press and hold in the liveman switch and then press and hold in the jog keys. Running an MML program does not depend on data transmission from the handheld pendant. The program will not stop if you release the liveman switch after the program starts.

The IMC 120 can accept the data you enter into the handheld pendant in jog (manual) or auto if you follow this sequence:

1. Release all keyboard keys
2. Press the liveman switch
3. Press the required keys

If you press more than one key at a time, neither key works. The IMC 120 ignores both keys.

2.4 Handheld Pendant Keys

Handheld pendant keys provide the letters, digits, and special character keys used to enter data into the prompts of the menus outlined in section 2.5.

The handheld pendant keys are also used for auto and jog operations from the pendant and for accessing status data related to the selected program and the axis.

2.4.1 Keys for Jog Operations

Jog keys are used to jog the axis and home the axis. Jog functions work only if the IMC 120 is in the jog (manual) mode. You can enable jog mode from the auto mode. See the JOGS key and chapter 5 for more information on jogs and homing.

The jog keys, listed below, are colored yellow on the handheld pendant. See figure 2.1 on page 2-2.

- JOGS key
- JOG- key
- JOG+ key

JOGS key
Press the <JOGS> key to enable jog operations. The control enables jog moves when an MML program is not running or is suspended and all programmed motion is complete.

If you try to switch from auto to jog mode when an MML program is executing, the handheld pendant displays the message: "CAN NOT SWITCH MODES", and the IMC 120 ignores the <JOGS> key.

JOG+ and JOG- Keys
The <JOG+> and <JOG-> keys move the axis in jog mode. Holding down the <JOG+> or <JOG-> keys jogs the axis in the chosen direction.

The <JOG+> or <JOG-> key must be held down during the entire jog operation. The axis decelerates to a stop if you release the <JOG+> or <JOG-> key during a jog move.

Either of the following conditions can abort a jog move:

- release the <JOG+> or <JOG-> key

- release the live man switch

WARNING: To guard against personal injury or damage to equipment be aware that if you interrupt an incremental jog, the portion of the increment the axis has moved is NOT remembered. Pressing the <JOG+> or <JOG-> key again repeats the move from the current axis position.

2.4.2 Keys for Starting Auto Operations

AUTO Key

The <AUTO> key switches the IMC 120 from jog mode to auto mode. Use auto mode to run MML programs and enable block transfers to and from the PLC. The discrete data transfers take place every coarse iteration in the auto or jog mode. See chapter 8 for more information on the auto mode.

SINGLE STEP Key

The <SINGLE STEP> key toggles the IMC 120 between continuous execution and single step execution of the MML program in auto mode. Check to see if the MML program is in single step or continuous by looking at the Status display of the Status menu.

If you compile the MML program without debug information, you cannot single step the program (the printout listing has step numbers but the downloaded program does not). The IMC 120 forces continuous execution and you cannot single step the program.

RUN Key

The <RUN> key is active only in auto mode. When you are in the auto mode and no MML program is currently running (end of program, abort, or power on has just taken place), the program you select starts from the beginning when you press <RUN>. If you press the <RUN> key without selecting a program, the handheld pendant displays the message "NO PROGRAM SELECTED".

If you press the <RUN> key to restart a suspended MML program (possibly by a HOLD or PAUSE) it continues running from the interruption point. Any suspended motion also continues from the interruption point.

2.4.3 Keys for Stopping Auto Motion

Use table 2.A with this section to help distinguish between the functions of the keys. See the IMC 120 Motion Control System Programming Manual for more information.

HOLD Key

The <HOLD> key stops motion immediately to a stop and stops the program after the current statement finishes. You can start the motion again by pressing <RUN>.

Important: Hold from the handheld pendant or the program does not enable jog mode (<JOGS> does). Current motion is not completed but only suspended. To enter jog mode after a hold, you must press the <ABORT MOVE> key to cancel the motion.

PAUSE KEY

The <PAUSE> key stops only the program after it finishes the current statement, motion continues until all planned and queued motion are complete.

ABORT MOVE Key

The <ABORT MOVE> key stops and cancels only the current move.

The <ABORT MOVE> key aborts any motion that is on hold.

Example:

If you are debugging a program and press the <HOLD> key to stop a motion, you can press the <ABORT MOVE> key to cancel that motion. Note that any motion in the programmed motion queue starts when you press the <RUN> key to continue running the program.

ABORT PROG key

The <ABORT PROG> key immediately stops and cancels the motion and the program. The program resets to start from the beginning.

Table 2.A
Affects of Starting and Stopping Program Interpreter and Motion Execution

Function	Effect On:				Resumed With:
	Program Execution	Motion in Programmed Motion Queue	Motion Stack	Motion in Planned Motion Queue	
Pendant HOLD	stopped	none	none	decelerates to stop	pendant RUN
Pendant PAUSE	stopped	none	none	none	pendant RUN
Pendant ABORT MOVE	none	none	none	decelerates to stop and thrown away	n/a
Pendant ABORT PROG	aborted	flushed	flushed	decelerates to stop and thrown away	pendant RUN

2.4.4 Miscellaneous Keys

Up And Down Arrow Keys

The up arrow <↑> and down arrow <↓> keys increase or decrease the speed that the axis moves by ten percent each time they are pressed. These keys are active in any mode and in any menu.

You can decrease the current speed to 0 or increase the current speed to a maximum of 127 percent of the programmed speed. The speed override cannot cause the maximum speed of the axis to exceed the maximum programmable speed set in AMP.

You can monitor the speed override value by looking at the Machine display of the Status menu.

Example:

If Current Speed is:	Then Resulting Speed will be:
40 rpm	52 rpm
100 rpm	127 rpm

Press the <↑> up arrow key 3 times to increase the current speed by 30 percent.

Notice that the maximum axis speed is limited to 127 percent of the programmed speed even if the speed override calls for a higher speed.

Function keys

The <F1>,<F2>,<F3>, and <F4> keys select specific displays from the menu you are in. See figure 2.2.

Example:

Press <F1> in the Jogs main menu to display the jog type menu. Press <F1> in the jog type menu to select continuous jog.

MORE key

The <MORE> key accesses data that continues onto another display. The multiple displays of the Status menu is an example of this. A flashing M in the upper right hand corner of the display tells you more displays are available.

LEFT ARROW key

Press either <SHIFT> key to enable the <←> left arrow function. The left arrow key functions as a delete key in the left direction. You can delete data any time before you press the <ENTER> key.

If you leave a prompt display that you typed data into but did not enter with the <ENTER> key, the typed in data is lost. See Data Entry in section 2.4.5 for more information on data entry.

2.4.5 Data Entry

Use the keys on the handheld pendant to enter data into the menu prompts on the displays. The cursor blinks at the point where you need to enter the data.

Alpha Entry

To enter alphabetic data:

	Press	Result
1.	left < SHIFT > twice	Enable continuous alpha data entry. The blinking cursor on the display changes to a square with a darkened upper left corner. The left < SHIFT > key enables the character in the upper left corner of the keys on the handheld pendant.
2.	< LETTER KEYS > *	fill in the prompts
3.	left < SHIFT >	Unlock continuous alpha entry. The cursor changes back to the blinking plain box for the unshifted state.
4.	< ENTER >	Enter the keyed in letters into the IMC 120.

Notes:

* Erase keyed in data by pressing the left arrow key for each character to be erased before you press the < ENTER > key. Also, changing to another menu will lose data that has been keyed in but not entered with the < ENTER > key.

The cursor changes with the active shift mode.

	Press	Cursor	Key Mode
	(unshifted)	▣	No Data Entry Mode
◻ SHIFT	left SHIFT key	◪	Alpha Entry Mode
Y ◻ SHIFT	right SHIFT key	◩	Numeric Entry Mode

Digit Entry

To enter numeric data:

Press	Result
1. right <SHIFT> twice	Enable continuous digit data entry. The blinking cursor on the display changes to a square with a darkened upper right corner.
2. <DIGIT KEYS> *	fill in the prompts
3. right <SHIFT>	Unlock continuous digit entry. The cursor changes back to the unshifted state.
4. <ENTER>	enter the keyed in numbers

Notes:
* Erase keyed in data by pressing the left arrow key for each character to be erased before you press the <ENTER> key. Also, changing to another menu will lose data that has been keyed in but not entered with the <ENTER> key.

TEACH POSN Key

The <TEACH POSN> key is used to initialize non-position variables including MOVE TO and MOVE AT SPEED, and to teach the position of the last MOVE TO statement completed.

To check the value of the variable, enter the variable name into the Modify Variable display of the Debug menu. See Teach Position menu in section 2.5.1 and Initializing Variables in chapter 6 for more information.

To initialize a variable -- The MML Program stops when it encounters an uninitialized variable in auto mode. Press the <TEACH POSN> key to display the Teach main menu. Follow the prompts to initialize the variable.

To teach the last MOVE TO position -- Select the program and run it in single step until it has executed the MOVE TO statement you want to teach. When the motion stops press the <JOGS> key and jog to the position where you want the MOVE TO statement to stop. Press the <TEACH POSN> key to display the Teach Position main menu. Follow the prompts to initialize the variable.

2.5 Displays Overview

The handheld pendant display is a 4 line by 16 character display that shows the error messages from the IMC 120 system or the menu you enabled.

Select any of the four major menus (Jogs, Status, Debug, and File Manager menus) from anywhere in any menu in auto or jog mode by pressing the key for that menu.

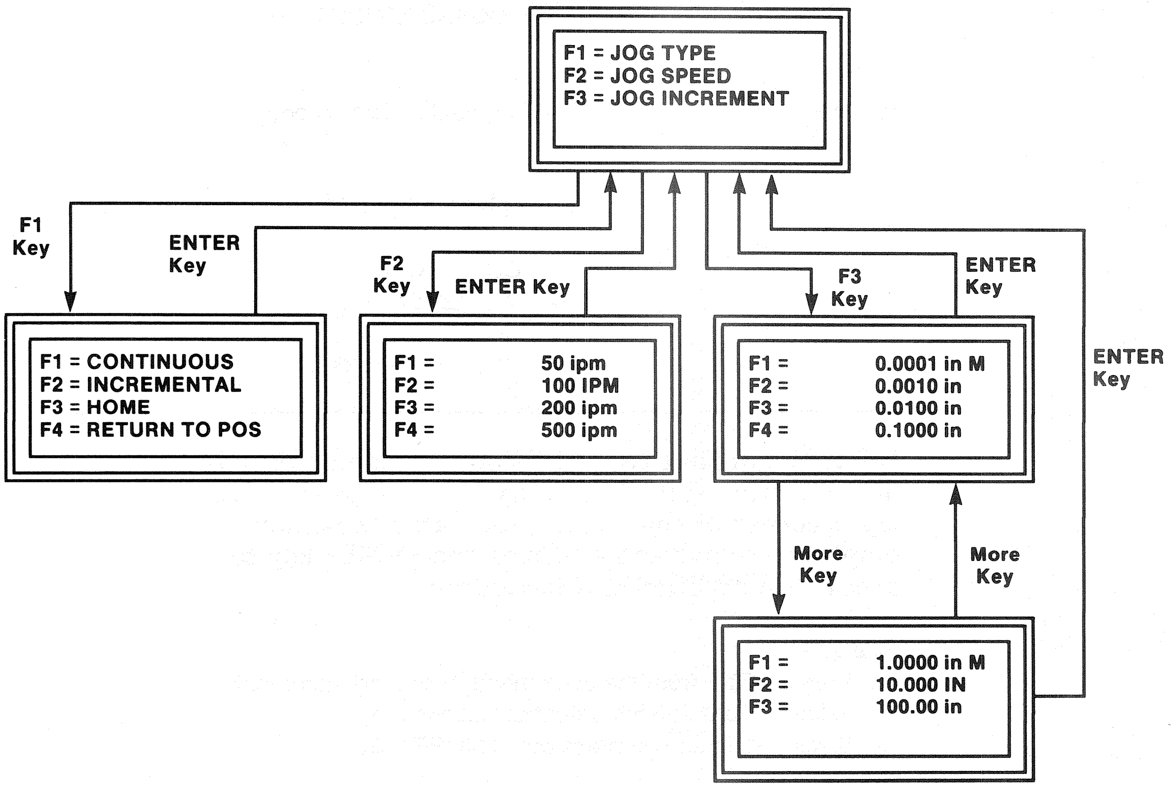
Key Pressed	Menu Displayed
<JOGS>	Jogs
<STATUS>	Status
<DEBUG>	Debug
<FILE MGR>	File Manager
<TEACH POSN>*	Teach Position

* The <TEACH POSN> key displays the Teach Position menu only if the MML program is running and encounters an uninitialized variable or you stop the program to modify the position value of the last move to statement run in the MML program. See section 2.5.1.

2.5.1 Menu Displays

Each box in figure 2.2 represents a menu display for the Jogs menu. This is an example to show the menu display. More information on this function is in later sections. The arrows between the menus point to the next menu that you see when you press the key next to the arrow.

Figure 2.2
Handheld Pendant Jogs Menu Display



15874

To access the menus on the handheld pendant press the key for the menu you want to see. All of the displays in a menu are related to each other. The main menu is the the most general. And each display is more detailed as you go down. For example:

Press	Result

1. <JOGS>	display the top (main) block of the Jog menu (figure 2.2).
	
	Example: Jogs Main Menu
	
	F1 = JOG TYPE
	F2 = JOG SPEED
	F3 = JOG INCREMENT

2. <F1>	display the Jog Type menu
	
	Example: Jog Type Menu
	
	F1 = CONTINUOUS
	F2 = INCREMENTAL
	F3 = HOME
	F4 = RETURN TO POSN

Information that cannot fit on a single display is divided into multiple displays. The M in the upper right corner of the display indicates that more displays are available. Press the <MORE> key to access the additional information.

Example:
Press <F3> from the Jogs menu to see jog increment values. More Jog Increment selections are displayed when you press the <MORE> key.

Moving Between Menu Displays

Table 2.B shows how to move through the display menus. This is general information that you can use in any menu display.

Table 2.B
How to Move Through the Display Menus

To Do This	*Press This*
Return to the main menu (top block) of the menu you are in	the key for that menu (JOGS, STATUS, FILE MGR, TEACH POSN, or DEBUG)
Back up a level in any menu	< ENTER > key
Page through multiple displays of a menu	< MORE > key

Jogs menu

Press the <JOGS> key to access the Jog menu on the handheld pendant. The <JOGS> key enables jog mode. The menu accesses the following functions.

- Jog Type
- Jog Speed
- Jog Increment

Important: The values for jog speeds and jog increments shown in figure 2.2 are for illustration only. Values for your system are selected through the AMP utility.

See chapter 5 for more information on jogs.

Status Menu

The <STATUS> key displays the Status menu on the handheld pendant. The function keys (<F1> through <F4>) and the <MORE> key traverse the Status menu to examine the following system functions.

- Program
- Programmed Motion
- Machine
- Mode of Operation
- Error Condition*
- Jog*

* Not all of the selections available in the main Status menu fit on one display. Press the <STATUS> key and then the <MORE> key to access this information.

Note that the Error Condition display shows only one error at a time. The flashing M is blanked out when the last error is displayed.

See chapter 4 for more Status menu information.

Debug Menu
The <DEBUG> key displays the Debug menu. The function keys (<F1> through <F4>) traverse the Debug menu to perform the following:

- examine and modify certain system variables and MML program variables
- set a breakpoint in the program
- perform integration operations

The Debug menus access the following functions.

- Select Program
- Open Loop/Normal Jog
- Invert D/A
- Reverse Main FB Phase
- Reverse Aux FB Phase
- Invert Probe
- Connect/Disconnect PLC
- Modify Breakpoint
- Clear Breakpoint
- Modify Variables
- Dry Run/Normal Run

The Debug menu is shown in figure 6.1. See chapter 6 for more information.

File Mgr Menu
Press the <FILE MGR> key to display the File Manager menus. The function keys (<F1> through <F4>) and the <MORE> key traverse the File Manager menu to perform system file operations. The menus access the following functions.

- Program Directory
- Select Program
- Copy Program
- Delete Program
- Rename Program*
- Program Information*
- Remove Debug Information*

* Not all of the selections available in the main File Manager menu fit on one display. Press the <FILE MGR> key and then the <MORE> key to access this information. See chapter 7 for more File Manager menu information.

Teach Posn menu

The Teach Posn Menu is displayed when you press the <TEACH POSN> key. This menu is used to teach an uninitialized variable or to teach the position of the last MOVE TO statement completed. See chapter 6 for more information.

uninitialized variable -- The MML program stops when it finds an uninitialized variable. You must either fill in the variable or abort the program. The program cannot run past an uninitialized variable.

Press the <TEACH POSN> key to display the Teach Position menu, then follow the prompts to initialize the variable. To check the value of the variable, enter the variable name into the Modify Variable display of the Debug menu.

Important: We do not recommend initializing non-position variables when the program is running.

the last MOVE TO position -- Run the program until it has executed the MOVE TO statement you want to teach. Jog to the position where you want the MOVE TO statement to stop. Go back to auto mode and teach the position. Check the value of the variable in the Modify Variable display of the Debug menu.

3.0 Chapter Overview

This chapter discusses the following:

- powering up the IMC 120 system
- connecting the handheld pendant
- using the Emergency Stop button
- using the liveman switch
- running the handheld pendant diagnostics

3.1 Powering Up the IMC 120 System With the Handheld Pendant

Before you turn the POWER switch ON at the power supply and press the Emergency Stop reset button, you should make sure that:

- the input power lines are wired correctly to the power supply

- the voltage is set correctly on the power supply (120VAC or 220VAC)

- all user power cables between the power supply and the IMC 120 modules (cat. no. 1771-CAS) are connected and routed appropriately

- all cables from IMC 120 modules to each termination panel are connected

- all wiring from each termination panel to drives, feedback devices, fast input and output devices, Emergency Stop string and Emergency Stop Reset button are connected properly

- the AMP is downloaded. You cannot reset an Emergency Stop unless the AMP (System Parameters) is downloaded.

- the MML Program is downloaded

3.1.1 Connect The Handheld Pendant

The pendant has a 25 foot RS-232 cable that carries data to and from the handheld pendant.

Plug the handheld pendant into the 9 pin D-Shell connector on the servo controller module. This connector has long Emergency Stop pins to make connection before the handheld Emergency Stop bypass becomes disengaged. The handheld pendant is part of the emergency stop circuit when it is connected to the servo controller module.

Press the Emergency Stop reset button when you disconnect the handheld pendant or the IMC 120 system will sense a break in the circuit and go into Emergency Stop. You do not need to press the Emergency Stop Reset button to plug in the handheld pendant.

3.1.2 Emergency Stop and Liveman Switch

The Emergency Stop pushbutton is a one inch diameter red mushroom type button used to signal the servo controller module of an emergency condition. To activate, push the button in. To deactivate, pull the button out and press the user supplied Emergency Stop reset button.

The liveman switch enables serial transmission from the handheld pendant to the IMC 120 servo controller module. When the liveman switch is released, transmission is terminated.

The procedure for transmitting data is:

1. Release the keyboard keys.

2. Press the liveman switch.

3. Press the keyboard keys.

Important: The liveman switch disables transmission when it is released even if a <JOG+> key is pressed.

Emergency Stop and Emergency Stop Reset

Plug in the hand held pendant into the IMC 120 servo controller module. Pull out the Emergency Stop button and then press and hold the Emergency Stop Reset pushbutton to make the system come out of Emergency Stop.

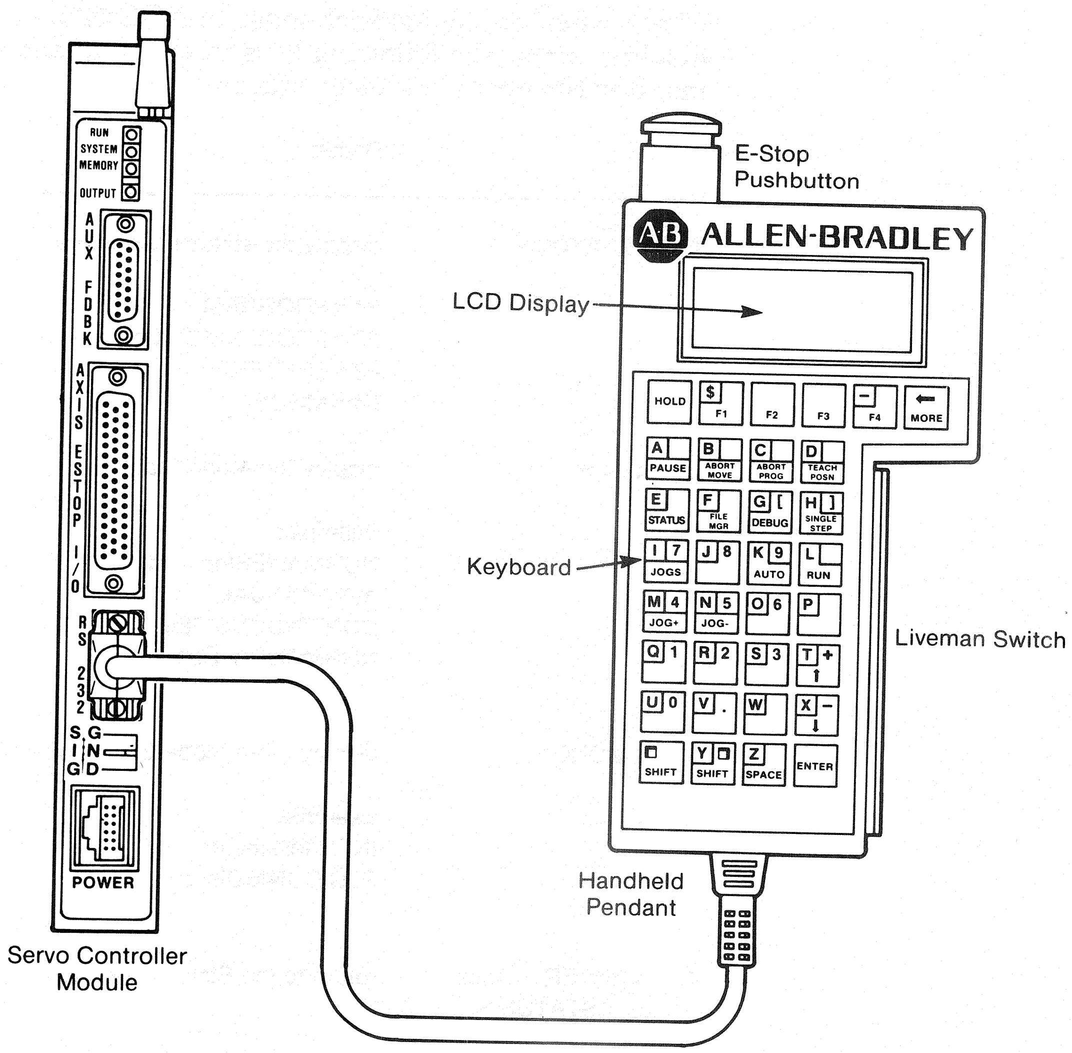
Check whether the Mode changes to NORMAL on the mode display. Press the following keys in this sequence to monitor the operating mode status.

Press	Result

1. <STATUS>	display the status main menu
	
	F1 = PROGRAM
	F2 = PROG'D MOTION
	F3 = MACHINE
	F4 = MODE

2. <F4>	display the Mode Status **Example:** NORMAL/EStop M Auto/MANUAL CONTINUOUS/Step NORMAL/Dry Run
3. <MORE>	display more Mode Status information **Example:** INCH/Millimeter PLC CONN/Disconn
4. <ENTER> twice or <STATUS>	return to the Status main menu
or <MORE>	return to the first page of Mode Status

Figure 3.1
Handheld Pendant Connected to Servo Controller Module RS-232 Connector



3.2 Diagnostics for the Handheld Pendant

The built-in diagnostic tests all keys, switches and buttons (except Emergency Stop), and the liquid crystal display on the hand held pendant. You can only run the diagnostics at power up.

The diagnostic executes the following tasks:

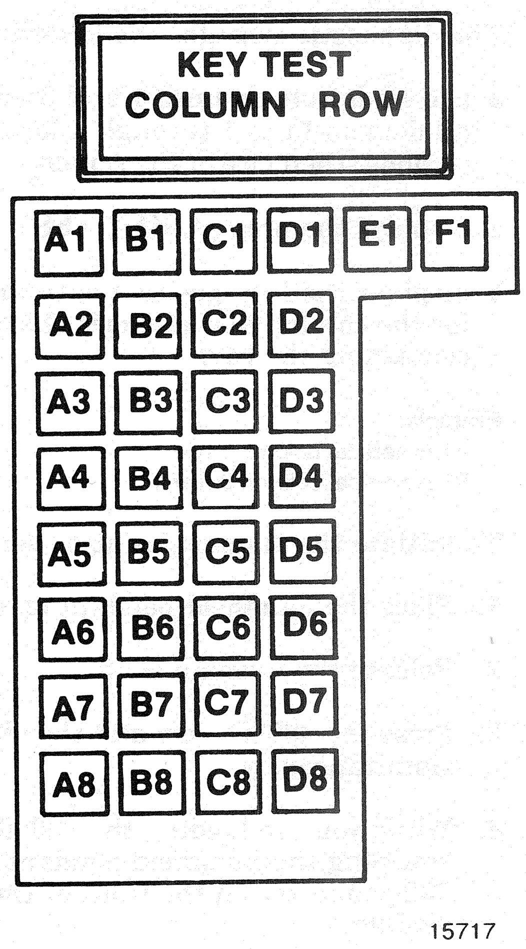
- activates dots on lines 1 and 3 and then activates all dots on lines 1 through 4 for a period of 5 seconds, then clears the screen
- displays the prompt, KEY TEST
- displays the keys position with an alpha character for the column and numeric character for the row containing the key.

Example:
A1 is vertical column 1 row 1,
F1 is vertical column 6 row 1

To initiate the diagnostics at power up:

1. Place the handheld pendant on a stable surface.
2. Release the liveman switch.
3. Press the `<HOLD>` key and the `<MORE>` key simultaneously.
4. While you are holding the `<HOLD>` key and the `<MORE>` key, plug the handheld pendant cable into the RS-232 connector on the front of the servo controller module.
5. Press the keys one at a time to test them. If you press more than one key at a time, an error is displayed. Figure 3.2 shows the alpha columns and numeric rows of each key.
6. Press the liveman switch to terminate the diagnostics.

Figure 3.2
Alpha Columns and Numeric Rows for the Handheld Pendant Keyboard Diagnostics



Important: The handheld pendant does not transmit characters to or respond to characters received from the RS-232 communications port while the diagnostics are running.

4.0 Chapter Overview

This chapter describes how to access and monitor the Status menu displays through the handheld pendant.

Press the <STATUS> key to display the Status main menu on the handheld pendant. Access the following system functions with the Status menu.

- Program
- Machine
- Jog*
- Programmed Motion
- Mode of Operation
- Error Condition*

* Not all of the selections available in the main Status menu fit on one display. Press the <STATUS> key and then the <MORE> key to access this information.

4.1 Monitoring Program Status

Function
The Program Status menu (figure 4.1 on page 4-3) shows the program running and the program condition. It contains:

- Program Name -- the identification you gave to the program when you created it on the Offline Development System.

- Program Step Number -- the command (program statement) currently running in the servo controller module.

- Program Status -- the condition of the program you selected, program status may be: RUNNING, NOT_RUN, or SUSPENDED.

- Program Wait List -- five possible messages represent signals the IMC 120 processor is waiting for. These messages display as long as the IMC 120 processor works and the condition is not cleared by the program or operator. The messages are listed on page 4-2.

The five wait list messages are:

1. **RUN** (Run) -- the program is ready to run and waiting for the operator to press the <RUN> key.

2. **DLY** (Delay) -- the program executed a delay command and is waiting for the command to time out.

   Example:
   DELAY 1000 holds up program execution for 1 second.

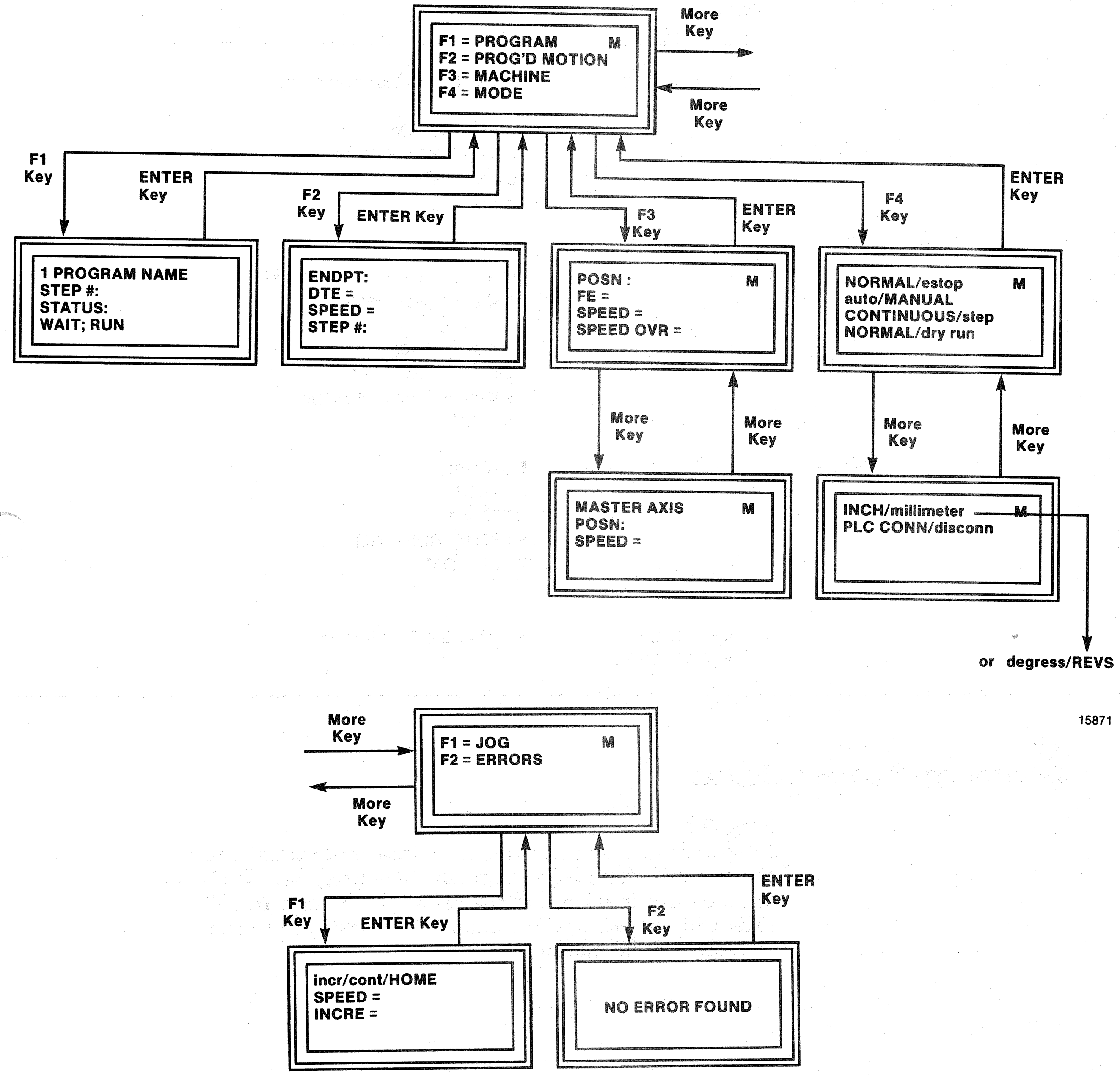
3. **COM** (Communication) -- a GET_PLC or PUT_PLC command was executed in the MML program and is held up until the PLC block transfer completes.

4. **MOT** (Motion Sync) -- the program executed a motion command and is waiting for a condition in that command to be met before the program can execute the next command.

   Example:
   Move to P1 and wait for the axis to move to within the fine tolerance value before executing the next block. (Wait for the termtype to be satisfied).

5. **TEACH** (Teach an Uninitialized Variable) -- the program is suspended waiting for the user to teach the encountered uninitialized variable.

Figure 4.1
IMC 120 Status Menu



Operation

Press the following keys in this sequence to monitor the program status.

Press	Result

1. <STATUS>	display the Status main menu
	
	F1 = PROGRAM
	F2 = PROG'D MOTION
	F3 = MACHINE
	F4 = MODE

2. <F1>	display the following information on the Program status menu. - the active program - the active step number - status of the active program - wait list **Example:** 01 TEST STEP # 1 STATUS: RUNNING WAIT: COM
3. <ENTER> or <STATUS>	return to the Status menu

4.2 Monitoring Program Motion

Function

Program Motion status displays data programmed into the current step of the running MML program. It shows the axis destination and the command being run. The IMC 120 automatically computes the distance to the endpoint of the motion.

The Programmed Motion Status menu contains:

- Endpoint of move -- the absolute dimension from the home position to where the current motion ends. Note that home position may not be zero. The home position value set in AMP may be modified by a preset or ORE.
- Distance to endpoint -- the dimension between the current axis position and the position where the current motion ends.
- Speed of move -- The rate of speed the axis is commanded to go.
- Step number of the programmed move -- the source code line number of the motion statement currently executing.

Operation

Press the following keys in this sequence to monitor program status.

Press	Result

1.<STATUS>	display the status main menu
	
	F1 = PROGRAM
	F2 = PROG'D MOTION
	F3 = MACHINE
	F4 = MODE

2. <F2>	display the following information on the program motion status menu. - endpoint of the programmed move - distance from the current location to the endpoint of the programmed move - current programmed axis speed - step number that is running **Example:** END PT: 1.20 DTE = 0.75 SPEED = 150 STEP#: 10
3. <ENTER> or or <STATUS>	return to the Status menu

4.3 Monitoring Machine Feedback

Function

The Machine Feedback menu shows machine axis status information. It contains:

- Machine Axis Position -- the current position of the axis. The absolute dimension from the home position where motion ended. Note that home position may not be zero. The home position value set in AMP may be modified by a preset or ORE.
- Machine Axis Following Error -- the amount of axis lag behind the command caused by mechanical frictions and gearing losses.
- Machine Axis Speed -- the actual speed the axis is moving (as compared to the speed the axis is commanded to go by the MML program).
- Speed Override Value -- the percent of reduction or increase of the programmed speed.

Speed Override value X programmed speed = Axis Speed

Operation

Press the following keys in this sequence to monitor the machine status.

Press	Result

1. < STATUS >	display the Status main menu
	
	F1 = PROGRAM
	F2 = PROG'D MOTION
	F3 = MACHINE
	F4 = MODE

2. < F3 >	display the machine status - current position of the axis - difference between the current axis position and the commanded axis position - actual axis speed - percent of speed override active Example: POSN: 2.50 M FE = .05 SPEED = 175 SPEED__OVR = 80
3. < MORE >	display the machine status on the auxiliary feedback input* of the servo controller module. - current position of the axis - actual master axis speed Example: MASTER AXIS M POSN: 2.50 SPEED = 175
4. < ENTER > or < STATUS >	return to the Status menu

Notes:
* This is the Master axis in ratioing applications. The auxiliary feedback input is used for ratioing two axis together. In ratioed motion, the IMC 120 is set up so that the motion of the axis which it controls follows the motion of another axis. The IMC 120 acts as a slave to this other axis, which is called the master.

4.4 Monitoring Operating Mode

Function

The Operating Mode menu shows currently active control modes. Capital letters indicate the active mode. The Operating Mode status menu contains:

- Normal/E-Stop -- In Normal the servo controller module operates in auto or jog mode.

  In E-Stop the servo controller module is disabled from operating in the auto or jog mode. The axis command is disabled, however the axis position is still monitored.

- Auto/Manual -- In manual you can jog/home the axis. Auto operationa are disabled in manual mode.

  In auto mode you can run the MML programs. Jog/home operations are disabled in auto.

  If you suspend the program execution to jog the axis, you need to select auto mode to resume the program execution. All condition handlers (program monitors) and fast input/output handlers (FINs/FOUTs) active when you suspended the program are enabled again. See the IMC 120 Motion Control System Programmers Manual for more information on condition handlers and fast interrupt statements.

- Continuous/Step -- In Continuous, the commands in the MML program run consecutively according to conditions in the program.

  In Step, the program is run one executable command at a time. The operator starts each command.

- Normal/Dry Run -- Normal enables MML program execution with axis movement enabled.

  Dry Run executes a program completely but disables all motion commands. The commands execute in the MML program but the axis doesn't move.

- Inch/Millimeter -- Inch sets the inch as the standard for computing and measuring distances along a linear axis. This selection is made in AMP with the units at power on parameter and can also be changed with the UNITS built-in procedure. You do not have to rehome the axis when you switch between Inch/Millimeter.

  Millimeter sets the millimeter as the standard for computing and measuring distances along a linear axis. This selection is made in AMP with the units at power on parameter and can also be changed with the UNITS built-in procedure. You do not have to rehome the axis when you switch between Inch/Millimeter.

- Degrees/Revolution -- Degrees sets degrees as the standard for computing and measuring distances for a rotary axis. This selection is made in AMP in the units at power on parameter and can also be changed with the UNITS built-in procedure. You do not have to rehome the axis when you switch between Degrees/Revolution.

  Revolution sets the revolution as the standard for computing and measuring distances for a rotary axis. This selection is made in AMP in the units at power on parameter and can also be changed with the UNITS built-in procedure. You do not have to rehome the axis when you switch between Degrees/Revolution.

- PLC Connect/Disconn (PLC Connected/Disconnected) -- PLC Connected logically connects the PLC to the IMC 120.

  PLC Disconnected logically disconnects the PLC from the IMC 120. The PLC block transfer output is accepted, filed, and ignored by the IMC 120 When the PLC is disconnected. Most discrete bits from the PLC are also ignored. This function is an optional step in the procedure to initialize program variables.

Operation

Each entry in this display is a copy of the status and not constantly refreshed as all the other displays are. The copy is recorded as the control accesses the display. The data display must be re-accessed periodically to be sure the display contains current data. This is the only display with this condition.

Note that active modes are displayed in upper case letters.

Press the following keys in this sequence to monitor the operating mode status.

Press	Result

1. <STATUS>	display the status main menu
	
	F1 = PROGRAM
	F2 = PROG'D MOTION
	F3 = MACHINE
	F4 = MODE

2. <F4>	display the Mode Status **Example:** NORMAL/EStop M Auto/MANUAL CONTINUOUS/Step NORMAL/Dry Run

3.	<MORE>	display more Mode Status information **Example:** INCH/Millimeter PLC CONN/Disconn
4.	<ENTER> twice or <STATUS>	return to the Status main menu
	or <MORE>	return to the first page of Mode Status

4.5 Monitoring Jog

Function
The Jogs Status display contains only the status of the Jog. Make changes to the Jog Status display in the Jogs menu. See chapter 5 for more information on jogging.

- INC/CONT/HOME: shows the currently active jog mode. The active mode is in upper case letters.
  - INC (Incremental) -- allows a jog only for the distance you specified.
  - CONT (Continuous) -- allows jog until you release the <JOG+> or <JOG-> key.
  - HOME allows you to jog only to the home position.
- SPEED -- shows the currently selected jog speed. You can modify Jog speed with the $SPEED__OVRD system variable.
- INCR (Increment) -- shows the currently selected jog increment.

Operation
Press the following keys in this sequence to monitor the jog status.

Press	Result

1. <STATUS>	display the status main menu
	
	F1 = PROGRAM
	F2 = PROG'D MOTION
	F3 = MACHINE
	F4 = MODE

2. <MORE>	display the More Status menu
3. <F1>	display the jog status **Example:** incr/CONT/home SPEED = 300 INCR = 10
4. <ENTER>	return to the More Status menu
or	
<ENTER> twice or <STATUS>	return to the Status menu

4.6 Monitoring System Errors

Function
You can display the hardware and software errors of the servo controller module with this selection. Errors are displayed one at a time in the order they occurred. The last error is indicated when the flashing M is blanked out. The possible error messages are listed in Appendix A.

Operation

Press the following keys in this sequence to display existing errors.

Press	Result

1. < STATUS >	display the Status main menu
	
	F1 = PROGRAM
	F2 = PROG'D MOTION
	F3 = MACHINE
	F4 = MODE

2. < MORE >	display the More Status menu
3. < F2 > *	display the error status **Example:** HARDWARE M ESTOP STEP#: 25
4. < MORE > **	display more errors **Example:** PLC IO REQUEST STEP#: 45
5. < ENTER >	return to the More Status menu
6. < STATUS > or < MORE >	display the Status menu

Notes:

* The flashing message "NO ERROR FOUND" is displayed if there is no error in the system.

** The M blanks out when you reach the bottom of the list. The < MORE > key stops working in this display at this point. Use the < ENTER > key or the hard keys representing the menu you want to see to exit.

5.0 Chapter Overview

This chapter explains how to go through the Jog menu display to jog the axis from the handheld pendant.

Use the Jog menu to select the type of jog, (continuous, incremental, home, or return to position) jog speed, and jog increment (for incremental jogs).

Jog moves are the only operations you can perform in manual mode to move the axis. Sometimes jog mode is referred to as manual mode.

5.1 Enabling Jog Mode

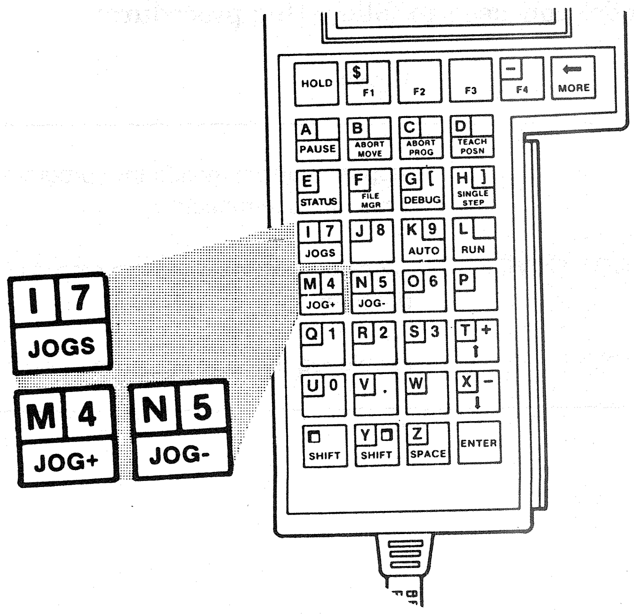
Function

Jog functions work only if the IMC 120 is in the jog (manual) mode.

The jog function allows you to move the axis from the handheld pendant without programming the PLC or the MML program.

You may jog the axis from the handheld pendant if the IMC 120 servo controller module is in jog mode, the servo is on, AMP has been downloaded from the ODS, and you are not in Emergency Stop.

The <JOG+> and <JOG-> keys are used for jogging the axis and homing the axis. The jog keys are colored yellow on the handheld pendant.



Operation

Begin the procedure for jogging the axis by selecting the Jog menu on the handheld pendant. Press <JOGS> to to enable jog operations and display the top level (main menu) of the Jogs menu from anywhere in any of the four major menus (Jog, Status, Debug and File Manager). Verify changes on the Jogs display of the Status menu.

The servo controller module powers up in jog mode but the operating mode may have been changed to run a program. You can change from auto to jog mode if:

- the program is not running or is suspended
- all motion is completed (reached the endpoint or is aborted) in a suspended program

Other conditions must also be met before you can jog the axis. If the axis doesn't jog and you know the program is not running and the motion is completed, check the following:

- System AMP parameters are downloaded from the ODS.
- IMC 120 is not in Emergency Stop.
- servo is on

5.1.1 Changing From Auto to Jog Mode

To change from auto to jog mode, press the <JOGS> key. If you get the message "CANT SWITCH MODES" then you are currently running a program and must cancel it first.

To access jog mode when you have a program running in auto mode, you need to follow this procedure:

Press	Result
1. < HOLD >	stop the current motion (the program doesn't stop, only the motion)
2. < ABORT MOVE >	cancel the current program step, to stop the program
3. < JOGS >	enable jog mode to allow jog moves

OR YOU CAN:

Press	Result
1. <HOLD>	stop any currently running motion.
2. <PAUSE>	stop the program and no more motion is started.
3. <JOGS>	enable jog mode to allow jog moves

OR YOU CAN:

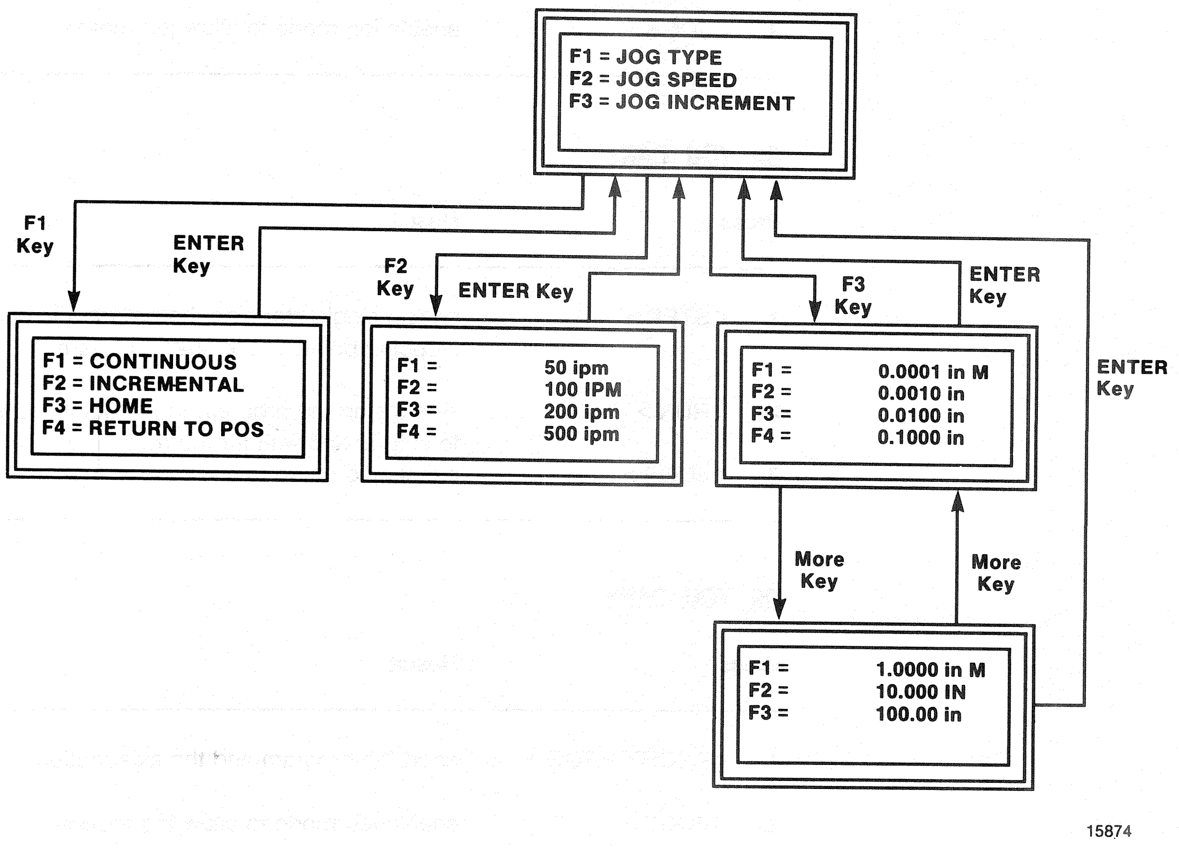
Press	Result
1. <STEP>	enable single-stepping the program to enable stopping points in the MML program
2. <RUN>	single-step the program to the point you want to stop at and wait for the motion to stop.
3. <JOGS>	enable jog mode to allow jog moves

OR YOU CAN:

Press	Result
1. <ABORT PROG>	abort the program and the axis motion.
2. <JOGS>	enable jog mode to allow jog moves

Important: Hold from the handheld pendant or from the program does not enable jog mode. Current motion is not completed but only suspended. The motion and the MML program must must stop before you can enable the jog function with the <JOGS> key.

Figure 5.1
IMC 120 Jogs Menu



Important: The values for jog speeds and jog increments shown here are for illustration only. These values are selected through AMP.

5.2 Setting Jog Type

Function
You need to select one of the four jog types from the Jog Type menu before you begin the jog move. Select a jog type with this procedure.

Operation

Press	Result
1. <JOGS>	display the Jog main menu.

2. <F1>	**Select Jog Type:** display the Jog Type menu.
	
	Example:
	F1 = CONTINUOUS
	F2 = incremental
	F3 = home
	F4 = return to pos

3. <F1>, or <F2>, or <F3>, or <F4>	select the desired jog type
4. <ENTER> or <JOGS>	return to the Jog main menu

Important: The active jog mode is displayed in uppercase letters on the Jogs display of the Status menu. The active jog type remains in effect until you choose another jog type from the jogs menu.

5.3 Setting Jog Speed

Function
Jog speed is the currently selected speed from the Jog Speed menu. The Jog Speed display contains the currently selected jog speed. The active jog speed remains in effect until you choose another speed from the jogs menu.

Important: Jog Speed is affected by the speed override value in the system variable $SPEED__OVR.

Operation

Press	Result
1. <JOGS>	display the Jog main menu.
2. <F2>	display the Jog Speed menu.

3. <F1>, <F2>, <F3>, or <F4>	select the desired jog speed
	Example:*
	F1 = 50 IPM
	F2 = 100 IPM
	F3 = 200 IPM
	F4 = 500 IPM

4. <ENTER>	return to the Jog main menu

Note:
* The values shown are examples only. The values you will see are the values set in AMP.

5.4 Performing Jog Moves

Function

The jog moves (continuous, incremental, home and return to position) begin when you press the <JOG+> or <JOG-> key. The jog move ends when the key is released, the liveman switch is released, or the intended destination is reached, whichever happens first.

Operation

Press the <JOG+> or <JOG-> key to select direction and start the jog move. The axis moves by the type, speed, and increment that you selected in the Jog menus.

Important: If you release the liveman switch during a jog move you need to follow these steps to restart the jog:

1. Release the <JOG+> or <JOG-> key.
2. Press the liveman switch again.
3. Press the <JOG+> or <JOG-> key again.

5.4.1 Continuous Jogging

Continuous jog permits jogging the axis as long as you press the <JOG+> or <JOG-> key with the liveman switch pressed, or until you reach an overtravel limit (if they are active).

Press	Result
	Select Jog Type:
1. <JOGS>	display the Jog main menu.

2. <F1>	display the Jog Type menu.
	
	Example:
	F1 = CONTINUOUS
	F2 = incremental
	F3 = home
	F4 = return to pos

3. <F1>	select CONTINUOUS jog type
4. <ENTER>	return to the Jog main menu
	Select Jog Speed:
5. <F2>	display the Jog Speed menu.

6. <F1>, <F2>, <F3>, or <F4>	select the desired jog speed
	
	Example:*
	F1 = 50 IPM
	F2 = 100 IPM
	F3 = 200 IPM
	F4 = 500 IPM

7. <ENTER>	return to the Jog main menu
8. <JOG + > or <JOG- >	jog to the desired position

Note:
* The values shown are examples only. The values you will see are the values set in AMP.

5.4.2 Incremental Jogging

Incremental jog permits jogging the axis by the amount of increment you select in the Jog Increment menu each time you press and hold the <JOG+> or <JOG-> key with the liveman switch pressed.

When you select Incremental jog you also need to select a jog increment from the Jog Increment display.

WARNING: To guard against personal injury or damage to equipment, be aware that the total distance of the incremental jog move repeats from the point the axis stopped each time the <JOG+> or <JOG-> key is pressed and held.

To jog the axis by an incremental distance follow this procedure:

Press	Result
	Select Jog Type:
1. < JOGS >	display the Jog main menu.

2. < F1 >	display the Jog Type menu.
	
	Example:
	F1 = continuous
	F2 = INCREMENTAL
	F3 = home
	F4 = return to pos

3. < F2 >	select incremental jog type

	Select Jog Speed:
4. <F2>	display the Jog Speed menu.

5. <F1>, <F2>, <F3>, or <F4>	select the desired jog speed
	
	Example:*
	F1 = 50 IPM
	F2 = 100 IPM
	F3 = 200 IPM
	F4 = 500 IPM

6. <ENTER>	return to the Jog main menu
	Select Jog Increment:
7. <F3>	display the Jog Increment menu.

8. <F1>, or<F2>, or <F3>, or <F4>	select the desired jog increment (press the <MORE> key for more distance selections).
	
	Example:**
	F1 = 0.0001 in
	F2 = 0.0010 IN
	F3 = 0.0100 in
	F4 = 0.1000 in
	<MORE> key
	F1 = 1.0000 in
	F2 = 10.000 in
	F3 = 100.00 in
	F4 = 1000.0 in

9. <ENTER>	return to the Jog main menu
10. <JOG+> or <JOG->	jog to the desired position

Notes:

* The values shown here are examples only. The values you see are the values set in AMP.

** The values shown are examples only. The values you will see are the values set in AMP.

5.4.3 Homing the Axis

Setting home position establishes a reference point on the axis to measure move lengths with. Homing places the axis at the origin or absolute zero position you set. Note that the home position may not be 0. The way the IMC 120 handles the homing function is determined by the Homing system parameters. See chapter 24 in the IMC 120 Motion Control System Programming Manual for more information on homing.

You need to hold the <JOG+> or <JOG-> pressed to complete the jog to home move.

There are two methods of homing an axis. You can home to a hardware switch or to the nearest resolver null or encoder marker. Select the home type in the Home to Switch or or Null/Marker system parameter in AMP.

> **Important:** If you home to a null or marker, at power up, before homing, you must jog the axis through at least one revolution of the feedback device. This allows the control to locate the null or marker location.

Home to a switch --

1. You jog the axis to the home switch at the speed you set in the Jog menu and in the direction on the jog button you press.

2. After the axis reaches the home switch the IMC 120 automatically sets up another move command to move the axis off the home position in the direction set in AMP in the Direction to Move Off Switch system parameter, at the speed set in AMP in the Speed of Move to Null/Marker parameter.

3. When the control finds the marker or null nearest to the switch in the direction you selected in step 2, it loads the current axis position into the value of the home position parameter in AMP. The IMC 120 modifies the home position with the value in the Home Calibration Value system parameter and the $PHASE system variable. The resulting value is the home position.

To home an axis to the home switch, follow this procedure.

Press	Result
1. <JOGS>	display the Jog main menu.

2. <F1>	**Select Jog Type:** display the Jog Type menu.
	
	Example:
	F1 = continuous
	F2 = incremental
	F3 = HOME
	F4 = return to pos

3. <F3>	select HOME jog type
4. <ENTER>	return to the Jog main menu
5. <F2>	**Select a Jog Speed:** display the Jog Speed menu.

6. <F1>, <F2>, <F3>, or <F4>	select the desired jog speed
	
	Example:*
	F1 = 50 IPM
	F2 = 100 IPM
	F3 = 200 IPM
	F4 = 500 IPM

7. <ENTER>	return to the Jog main menu
8. <JOG+> or <JOG->	jog to the home position switch.**

Notes:
* The values shown are examples only. The values you will see are the values set in AMP.

** When you jog to home, the IMC 120:
1. moves from home switch to home position in the direction selected in Direction to Move Off Home system parameter.
2. moves from home switch to home position at the speed selected in the Speed to Move to Null/Marker system parameter in AMP.
3. loads current position into Home Position system parameter
4. modifies home position with the Home Calibration Value system parameter and the $PHASE system variable. The message HOME SUCCESSFULL is displayed on the handheld pendant.

Home to the nearest resolver null or encoder marker -- When you are homing to a resolver null or encoder marker, the servo controller moves to the nearest null or marker regardless of the sign on the jog key you press. The axis homes at the value in the AMP system parameter Speed of Move to Null/Marker.

When the control finds the nearest marker or null, it loads the position into the Home Position Value system parameter in AMP. The IMC 120 modifies the home position with the Home Calibration Value system parameter and the $PHASE system variable. The resulting value is the home position.

To home an axis to the nearest resolver null or encoder marker follow this procedure:

Press	Result
1. <JOGS>	display the Jog main menu.
2. <F1>	**Select Jog Type:** display the Jog Type menu.

	Example:
	F1 = continuous
	F2 = incremental
	F3 = HOME
	F4 = return to pos

3. <F3>	select HOME jog type
4. <ENTER>	return to the Jog main menu
5. <JOG+> or <JOG->	jog to the home position *

Notes:
* When you jog to home, the IMC 120:
1. moves to the nearest null or marker (home position). The IMC 120 moves to home position at the speed selected in the Speed to Move to Null/Marker system parameter in AMP.
2. loads current position into Home Position system parameter
3. Modifies home position with the Home Calibration Value system parameter and the $PHASE system variable. The message HOME SUCCESSFULL is displayed on the handheld pendant.

5.4.4 Returning to Position

Return to Position moves the axis back to the position where you exited the auto mode to jog the axis. This procedure shows how to stop the program and motion and restart them after the jog move.

Press	Result
1. <HOLD>	Stops any currently running motion.
2. <PAUSE>	Stops the program and no more motion is started.
3. <JOGS>	Changes modes from auto to jog mode. The axis location is stored by the IMC 120. The Jog main menu is displayed.
	Select jog type:
4. <F1>	select Jog Type display
5. <F1>, <F2> or <F3>	select continuous, incremental or home jog
	Select Jog Speed:
6. <F2>	display the Jog Speed menu.
7. <F1>, <F2>, <F3>, or <F4>	select the desired jog speed
8. <ENTER>	return to the Jog main menu
9. <JOG+> or <JOG->	Jog the axis in the desired direction with the <JOG+> <JOG-> keys.
10. <JOGS>	Display the Jog main menu
	Select jog type:
11. <F1>	select Jog Type display
12. <F4>	Select RETURN TO POSITION to return to where the axis was when you stopped the program.
13. <AUTO>	Return the IMC 120 to the auto mode.
14. <RUN>	Restart the program from the command and position where it was stopped.

5.5 Monitoring Jog Status

Function
The status of the Jogs menu is shown in the Jogs menu or in the Jogs display of the Status menu. The active mode is in upper case letters.

Operation
Changes to the Jog Status menu display (section 4.5) are made from the Jogs menu.

5.6 Axis Jog Alternative

Function
The axis may also be jogged from the PLC if the IMC 120 servo controller module is in auto mode.

Operation
Jogging from the PLC follows the same rules as jogging from the pendant. Jogging from the PLC is explained in the IMC 120 Motion Control System Programming Manual.

6.0 Chapter Overview

This chapter discusses how to use the Debug menu to debug an MML program.

Important: You cannot add or delete sections of an MML program or correct logic errors from the handheld pendant. To modify a program you must edit the MML source program on the ODS development system, compile it, download it, and re-test it.

Debug tools provided in the IMC 120 include initializing and modifying variables, clearing breakpoints and dry running the MML program.

The Debug menu provides these functions:

- Select Program
- Open Loop/Normal Jog
- Dry Run/Normal Run
- Invert D/A
- Invert Probe
- Reverse FB Phase
- Connect/Disconnect PLC
- Modify Breakpoint
- Clear Breakpoint
- Modify Variables
- Reverse Aux Feedback

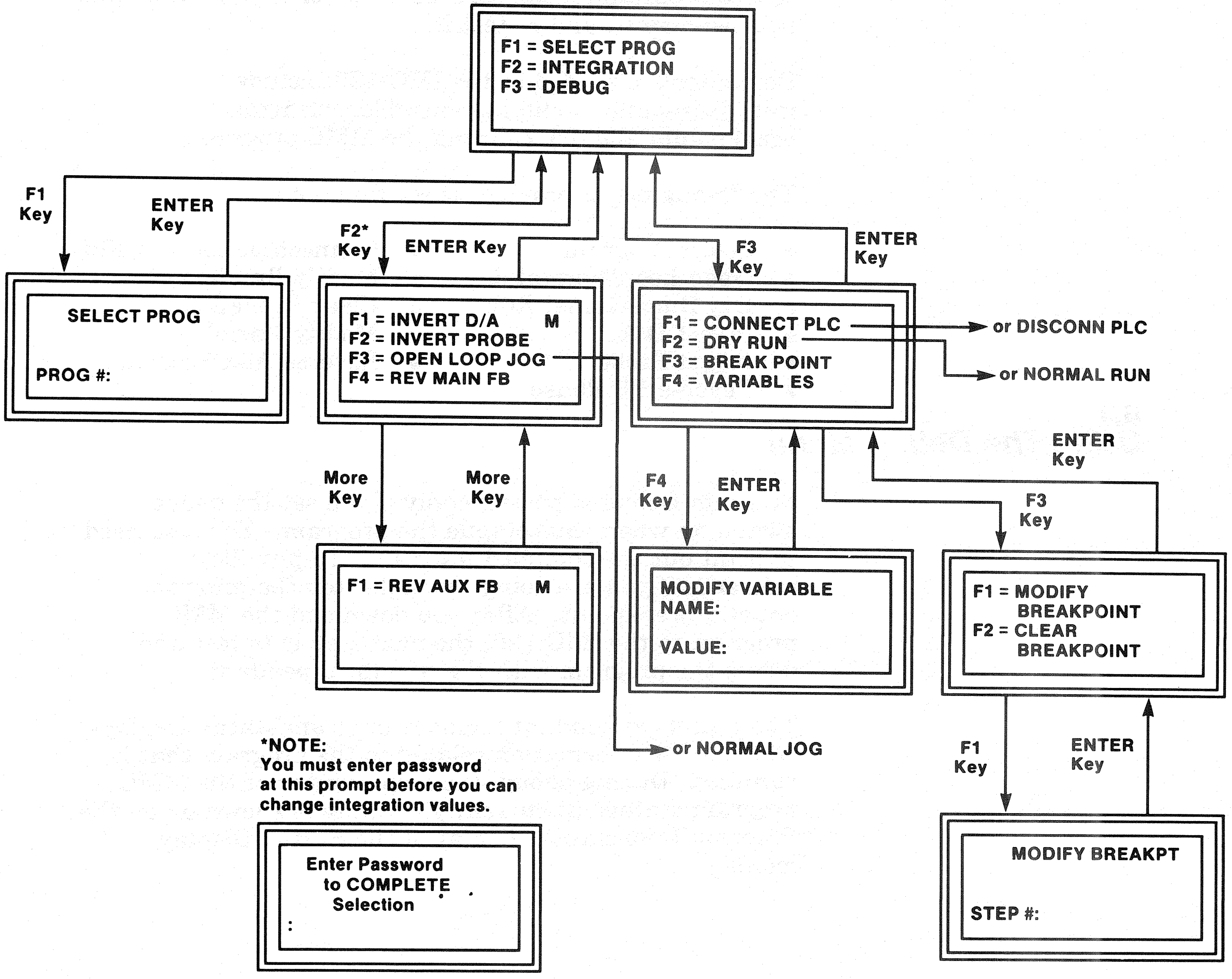
6.1 Using The Debug Menu

Program Debug is possible only if you set the debug switch on when you compile the program. You also need to print out the compiled program. To provide a reference if you are going to single step the program or set a breakpoint. After you download the MML program to the IMC 120, the next step is to test and debug the program with the handheld pendant.

The handheld pendant includes program status displays, which show information related to the program that is running. During debug, the step number of the MML program statement currently running is shown on on the Program Information display of the Status Display menu.

Figure 6.1 shows the Debug menus displays together to help you see the steps needed to access a specific menu.

Figure 6.1
IMC 120 Debug Menu Display



6.2 *Selecting a Program*

The following procedure describes how to select a program from the Select Program display.

	Press	Result

1.	< DEBUG >	display the Debug main menu
		
		F1 = SELECT PROG
		F2 = INTEGRATION
		F3 = DEBUG

2.	< F1 >	select the Select Prog display. SELECT PROG PROG #:
	at the PROG # prompt	
3.	right < SHIFT > key twice	enable number entry
4.	< NUMBER KEYS >	type in the number identifier of the program you want to make active
5.	<←> left arrow	clear errors from the input field before you press < ENTER >
6.	right < SHIFT > key once	disable number entry
7.	< ENTER >	enter the number and select the program and exit this display

6.3 Integrating The Axis

Important: The integration functions listed below are intended for use only during integration of the axis. Do not make changes to these functions after integration is complete. See the IMC 120 Motion Control System Installation Manual for more information on integrating the axis.

Press	Result
1. <DEBUG>	display the Debug main menu

2. <F2>*	select the Integration display.
	
	F1 = INVERT D/A M
	F2 = INVERT PROBE
	F3 = OPEN LOOP JOG/NORMAL JOG
	F4 = REV MAIN FB
	
	Press <MORE> key
	
	F1 = REV AUX FB M

Note:
* <F3> is either OPEN LOOP JOG or NORMAL JOG. This line of the display toggles to the active state of jog.

INVERT D/A -- Press <F1> to reverse the polarity of the digital to analog command signal to the axis.

INVERT PROBE -- Press <F2> to invert the polarity of the probe when it is actuated.

OPEN LOOP JOG/NORMAL JOG -- Press <F3> to jog with feedback from the axis (NORMAL JOG) or without feedback from the axis (OPEN LOOP JOG).

Normal jog uses feedback from the axis. You need to be in normal jog when you jog. The selection that you see is the mode the control is not in. If OPEN LOOP JOG is the selection available, the control is in normal jog. This is the mode you want unless you are integrating the axis.

REV MAIN FB -- Press <F4> to reverse the phase of the feedback signal from the axis controlled by the IMC 120.

REV AUX FB -- Press <MORE> and <F1> to reverse the linear touch probe, or or absolute feedback, or phase of the feedback signal connected to the auxilliary feedback connector (the master axis in ratio applications).

Caution: Changing the integration parameters may cause equipment damage. Do not change the integration parameters unless you have been trained to integrate systems. If the system has been integrated and is running correctly you do not need to change the integration parameters.

- Changing INVERT D/A may cause axis motion in an unanticipated direction.
- Changing OPEN LOOP JOG/NORMAL JOG may cause the axis to run away or run to the travel limit. OPEN LOOP JOG is used for jogging with no feedback from the axis.
- Changing REV MAIN FB may cause axis motion to occur in an unanticipated direction.
- Changing REV AUX FB may cause the IMC 120 to interpret the axis direction incorrectly.

6.4 Debug MML Programs

Figure 6.1 shows the Debug displays to show you the steps needed to access a specific menu. The Debug menu provides the following functions:

- Connect/Disconnect PLC
- Dry Run/Normal Run
- Modify Breakpoint
- Clear Breakpoint
- Modify Variables

Press	Result

1. <DEBUG>	display the Debug main menu
	
	F1 = CONNECT PLC/DISCONN PLC*
	F2 = DRY RUN/NORMAL RUN*
	F3 = BREAKPOINT
	F4 = VARIABLES

2. <F1>, or <F2>, <F3>, or <F4>	enable the function you want.

Notes:
* The <F1> line is either CONNECT PLC or DISCONN PLC. This line toggles to display the available selection, which is the inactive state.

The <F2> line is either DRY RUN or NORMAL RUN. This line toggles to display the available selection which is the inactive state.

6.4.1 Connect/Disconnect PLC

Function
During the debug phase, the PLC (and the system the PLC is in) can be logically disconnected from the IMC 120. System control and program control commands like "PROGRAM START" and "PROGRAM STOP", which can be initiated from the PLC, are ignored if the PLC is disconnected. Disconnecting the PLC inhibits the host PLC from starting and stopping an MML program while you are debugging a program.

Caution: To guard against possible damage to equipment do not disconnect the PLC from the IMC 120 during normal operating mode.

Variable information passed from the PLC, and all information going from the IMC 120 to the PLC continues to be passed even when the PLC is logically disconnected.

See the $DISABLE__PLC system variable for a summary of all PLC commands that are disabled. IMC 120 to PLC single transfer is unchanged. See the IMC 120 Motion Control System Programming Manual.

Operation

	Press	Result
1.	<DEBUG>	display the Debug main menu
2.	<F3>	select Debug display
3.	<F1>	in the Debug display to disconnect the PLC.
4.	<F1>	again to reconnect the PLC to IMC 120 communication.

Dry Run/Normal Run -- Dry run allows you to run the MML program without enabling and moving the axis.

Normal run allows the MML program to move the axis through the IMC 120. The PLC may cause axis movement through the MML program.

	Press	Result
1.	<DEBUG>	display the Debug main menu
2.	<F3>	select Debug display
3.	<F2>	select Dry Run mode
	or	
	<F2>	select Normal run mode

6.5 Modify/Clear a Breakpoint

Function

A breakpoint allows you to stop the program at a specific step number when in continuous run mode. The compiler in the ODS indicates the step numbers where possible break points can be set.

The ODS notes possible breakpoints and puts asterisks in the program printout at these steps if the Debug option is selected when you compiled the program. Step numbers marked by "*" are legal places to put a breakpoint. Operator intervention is required to restart a program that has stopped at a breakpoint.

Only one active breakpoint is allowed per MML program. You can set as many legal breakpoints as you want, but the program will only stop at the last one set. Each time you enter a new breakpoint it replaces the previous one.

The program should execute each statement up to, but not including, the breakpoint statement. The variables may be examined to determine correctness. If you set a breakpoint and the breakpoint was not encountered when you ran the program, then the step number was not executed and the control of the program went to another part of the program.

Operation

Clear a Breakpoint -- Select and run the program until the program reaches the breakpoint. Then press <F2> from the Clear Breakpoint display of the Debug menu.

Enter or Clear a Breakpoint -- Get the step number from the compiler listing. Valid steps for breakpoints are indicated with an asterisk.

	Press	Result
1.	< DEBUG >	display the Main Debug menu
2.	< F3 >	select the Debug display
3.	< F3 >	select the Modify/Clear Breakpoint display
4.	< F2 >	select the Clear Breakpoint display

Modify a Breakpoint -- Modify breakpoint allows you to select the step number where you want the program to stop. The program resumes when you press the <RUN> key.

Modify a breakpoint with this procedure:

Press	Result
1. < DEBUG.	display the Debug main menu
2. < F3 >	select the Debug display
3. < F3 >	select the Modify/Clear Breakpoint display
4. < F1 >	select the Modify Breakpoint display
at the STEP # prompt	
5. right < SHIFT > key twice	enable number entry
6. < NUMBER KEYS >	type in the number identifier of the step you want to stop on
7. <←> left arrow	clear display errors before you press < ENTER >
8. right < SHIFT > key once	disable number entry
9. < ENTER >	enter the breakpoint step number, exit this display
10. < DEBUG >	display the Debug main menu

6.6 Modifying Variables

Function
The Modify Variable menu display allows you to change the value of a system variable or program variable. The new variable value replaces the original one in the system or the program. The changes do not update the original original program loaded into the ODS. You must upload the program to make a record of the changes on the ODS. See the IMS 120 Motion Control System Programming Manual for more information.

The handheld pendant display shows the value of the system variable if the value is within the specifications of the option of the Debug menu. The following list describes the handheld pendant display specifications.

- The numbers entered into an MML program in scientific notation are converted to base 10 to be displayed..
- variable display space is 10 characters wide
- smallest negetive number displayed is -0.0000
- smallest positive number displayed is 0.00000
- largest positive number that can be displayed is 999999999
- largest negetive number that can be displayed is -99999999
- largest legal positive number is 2147483647
- largest legal negetive number is -214748364
- Smallest fractional part of a number is 1E-6

To examine the value of a variable, select the name from the program listing printed by the compiler and enter the variable name into the Modify Variable display of the Debug menu on the handheld pendant. You see the name and the current value of the variable if you entered the name correctly.

To alter a variable, first look up the variable in Appendix B to see if it is a read/write or read only variable. Read/write variables can be altered through the handheld pendant, read only variables cannot.

You can turn FOUTs and DOUTs on and off through the Modify Variable display to test them. See the IMC 120 Installation Manual for the procedure.

Operation

To modify a variable :

	Press	Result
1.	<DEBUG>	display the Debug main menu
2.	<F3>	select the Debug display
3.	<F4>	select the Modify Variable display MODIFY VARIABLE NAME: $SPEED
	at the NAME: prompt	
4.	left <SHIFT> key twice	enable letter entry
5.	<LETTER KEYS> *	type in the name of the variable you want to modify
6.	<←> left arrow	clear input errors before you press <ENTER>
7.	left <SHIFT> key once	disable letter entry
8.	<ENTER>	call up the variable. This is an example of the variable value prompt. MODIFY VARIABLE NAME: $SPEED VALUE:

	If no variable is found to match the name you typed in, this message is displayed: VARIABLE NAME NOT FOUND STEP #
at the VALUE: prompt	
9. right <SHIFT> key twice	enable number entry
10. <NUMBER KEYS> *	Type in the value you want for the variable you are modifying.
11. <←> left arrow	clear errors from the input field before you press <ENTER>
12. right <SHIFT> key once	disable number entry
13. <ENTER>	enter the value for the variable
14. <DEBUG>	return to the Debug main menu

Notes:

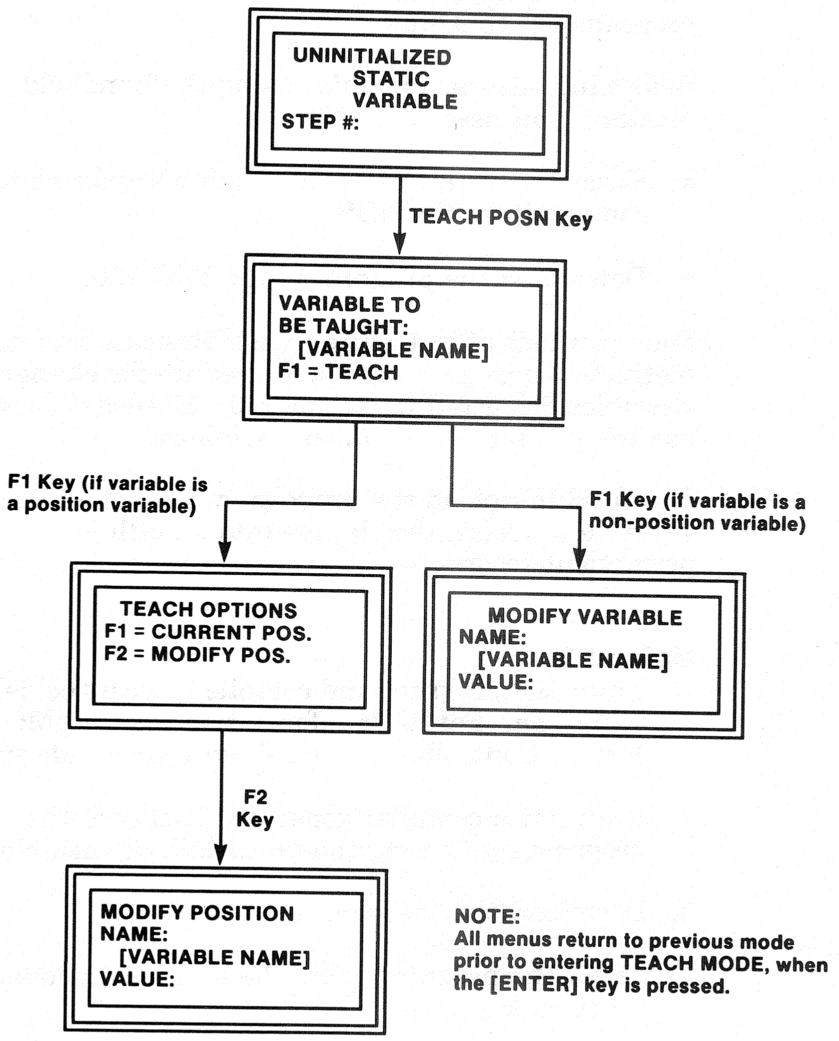
* If you enter a dollar sign prefix to the variable, the software knows it is a system variable.

If you enter a variable with no dollar sign prefix, the software knows it is an MML program variable.

6.7 Using The Teach Menu

The Teach Posn Menu is displayed when you press the <TEACH POSN> key. This menu is accessable when the program encounters an uninitialized variable or when you stop the program after a MOVE TO statement to teach the last completed MOVE TO statement position.

Figure 6.2
IMC 120 Handheld Pendant Teach Menu Display



15872

6.8 Uninitialized Variables

Function

Each time a program attempts to execute an uninitialized variable the program execution is suspended and you are prompted to initialize the variable. After the variable value is entered the program can continue.

Before initializing variables using the handheld pendant, you need to:

- Enter the source MML program with the editor and compile it on the ODS.
- Down load the program to the IMC 120.

Four methods of initializing variables can be used. Method 1 through 3 use the procedure for changing variables discussed in section 6.9. Method 4 does not use the pendant to initialize variables.

You need to upload the program after initializing the variable to record the change into an offline permanent record.

Method 1

1. Enter the program and compile it with the debug switch on. For more information see the IMC 120 Motion Controller System Programmers Manual.
2. Print out a compiler generated listing of the program containing the uninitialized variables.
3. Download the program to the IMC 120.
4. Select the program using the handheld pendant function keys.
5. Using the listing, identify uninitialized variables and follow the procedure in section 6.9.

Method 2

1. Enter and compile the program.
2. Print out a compiler generated listing of the program containing the uninitialized variables.
3. Download the program into the IMC 120 servo controller module.
4. Select the program using the handheld pendant function keys.
5. Program a break point on a line in the program where you want to initialize the variable (see section 6.5).
6. Press <RUN> to execute the program up to the break point you placed in a line prior to the uninitialized variable.
7. Use the listing and follow the procedure in section 6.9.

Method 3

1. Write the program with uninitialized variables.
2. Enter, compile, and download the program.
3. Select the program using the handheld pendant function keys.
4. Run the program.
5. When a line is encountered in the program containing an uninitialized variable, the program is paused waiting for that variable to be given a value.
6. Follow the procedure in section 6.9 as you encounter each uninitialized variable.

Method 4

1. Initialize all variables by entering values for them when you enter the program into the Offline Development System.
2. Compile the program, print out a listing (if desired), and download it to the IMC 120 servo controller module.
3. Run the program (in single step mode if desired).

6.9 Initializing Variables

The IMC 120 stops MML program execution when it finds an uninitialized variable. When this happens, this message is displayed:

```
UNINITIALIZED
   STATIC
  VARIABLE
STEP #: XX
```

You must either fill in the variable or abort the program. The program cannot run past an uninitialized variable.

Press the <TEACH POSN> key to display the Teach Position menu and the name of the uninitialized variable.

The menu displayed depends on the type of variable. If the uninitialized variable is a position variable the menu on the left of figure 6.2 is displayed. If the uninitialized variable is other than a position variable the menu on the right of figure 6.2 is displayed.

	Press	Result
1.	< TEACH POSN >	display the Teach main menu. The main menu of figure 6.2 shows the uninitialized variable prompt on the pendant.
2.	< F1 >	select the uninitialized variable display

Follow the prompts to initialize the variable. To check the value of the variable, enter the variable name into the Modify Variable display of the Debug menu.

6.9.1 Initializing Non-position Variables

Important: Variables (other than position variables) should be initialized when you develop the MML program. We do not recommend initializing non-position variables when the program is running.

If the uninitialized variable is a non-position variable the following prompt is displayed.

```
   MODIFY VARIABLE
NAME:
<variable name>
VALUE:
```

Fill in this prompt with the following procedure:

Press	Result
at the VALUE: prompt	
1. right <SHIFT> key twice	enable number entry
2. <NUMBER KEYS>	type in the value for the variable
3. <←> left arrow	clear the input field before you press <ENTER>
4. right <SHIFT> key once	disable number entry
5. <ENTER>	enter the number and return you to the menu that you were on before entering the Teach menu.

6.9.2 initializing Position Variables

If the uninitialized variable is a position variable the following prompt is displayed:

```
TEACH OPTIONS
F1 = Current Pos
F2 = Modify Pos
```

Teach an uninitialized position variable by one of the two methods in this display:

Teach Position Variable Method 1

Press	Result
1. <F1>	enter the current position of the axis and return to the display you were on before encountering the uninitialized variable.
2. <F2>	display the following prompt: MODIFY POSITION NAME: <VARIABLE NAME> VALUE:
at the VALUE: prompt	
3. right <SHIFT> key twice	enable number entry
4. <NUMBER KEYS>	type in the value for the variable
5. <⟵> left arrow	clear errors from the input field before you press <ENTER>
6. right <SHIFT> key once	disable number entry
7. <ENTER>	enter the value and return to the menu that you were on before entering the Teach menu.

Teach Position Variable Method 2

If the variable is a position variable, you may change to manual mode and jog the axis to the desired position for that position variable, then :

Press	Result
1. <TEACH POSN>	return to the Teach Position menu
2. <F1>	store the current axis position in the uninitialized position variable and continue program execution.

6.9.3 Teaching a MOVE TO Variable

Teaching the MOVE TO variable is intended let you fine tune the programmed end point of an axis move to a specific application. You can modify the global (static) position variable in the program with this procedure. When you modify any MOVE TO position variable, the program in the IMC 120 is changed. You have to upload the program to the ODS if you want a copy.

Single step the MML program until it has executed the MOVE TO statement you want to teach. When the motion stops, change to jog mode and jog to the position where you want the MOVE TO statement to stop, then :

	Press	Result
1.	<TEACH POSN>	display the Teach Position menu
2.	<F1>	key to store the current axis position in the uninitialized position variable and continue program execution.

The IMC 120 looks for uninitialized variables before unitialized MOVE TO position variables. If the IMC 120 does not find an uninitialized variable in the MML program statement, it looks for an ininitialized MOVE TO position variable.

If the IMC 120 doesn't find either one, it searches the next program statement for an uninitialized variable and then for an initialized MOVE TO position variables. This search continues to the end of the program.

7.0 Chapter Overview

This chapter discusses how to use the IMC 120 file management function to change MML program file information. Functions available for changing MML program files are:

- Select a program
- Copy a program
- Delete a program
- Rename a program
- monitor program and machine status

You also can maximize program storage by removing the debug information from the MML program if you compiled the MML program with the debug information function active.

7.1 Using the File Manager Menu

The File Manager displays on the handheld pendant are:

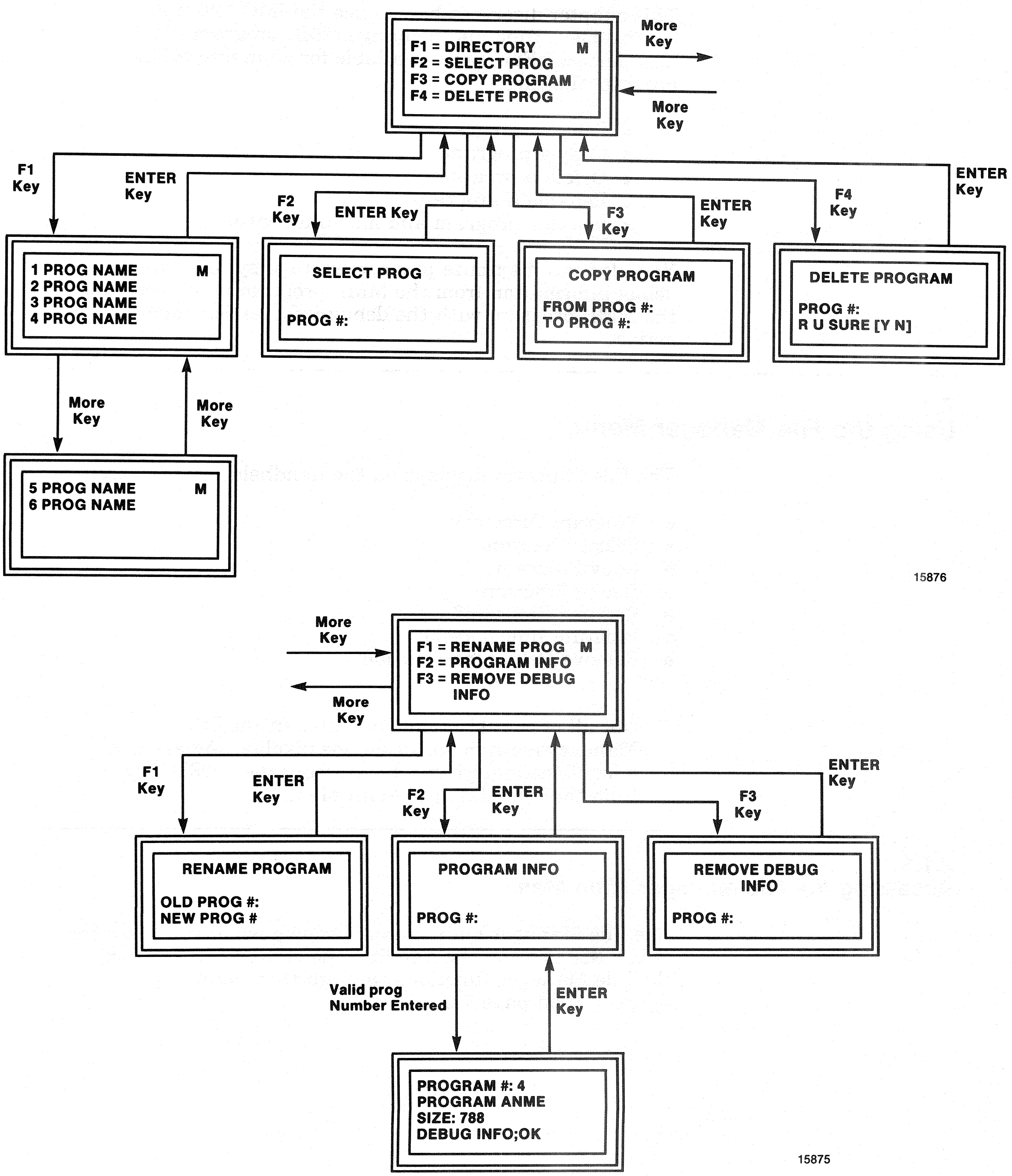
- Program Directory
- Select Program
- Copy Program
- Delete Program
- Rename Program*
- Program Information*
- Remove Debug Information*

* Not all of the selections available in the File Manager main menu fit on one display. Access this part of the main menu by pressing the <MORE> key from the File Manager Main Menu.

7.1.1 Accessing the File Manager Main Menu

The File Manager main menu is shown when you press the <FILE MGR> key on the handheld pendant. Access all of the File Manager functions through this menu. See figure 7.1. on page 7-2.

Figure 7.1
IMC 120 File Manager Menu



7.1.2 Locating a Program in the Directory

Function
The program Directory lists the MML programs in memory (31 programs maximum). Follow these steps to locate a program in the Directory of the File Manager menu.

Operation

	Press	Result
1.	<FILE MGR>	display the File Manager menu
2.	<F1>	display the first four programs stored in the Directory.
3.	<MORE>	display the next four programs in memory.
4.	<MORE> or	follow the list to the end and press <MORE> again to display the first four programs
5.	<ENTER> or <FILE MGR>	return to the File Manager main menu.

7.1.3 Selecting a Program

Function
Select Program lets you choose the active (default) MML program from the programs stored in the IMC 120. Choose a program to be active with this proceedure:

Operation

	Press	Result
1.	<FILE MGR>	display the File Manager menu
2.	<F2>	display the Select Program menu and the prompt: SELECT PROG PROG #:
at the PROG # prompt		
3.	right <SHIFT> key twice	enable number entry
4.	<NUMBER KEYS>	type in the number identifier of the program you want to run
5.	<←> left arrow	clear errors from the input field before you press <ENTER>
6.	right <SHIFT> key once	disable number entry
7.	<ENTER>	enter the number and select the program. When you enter a valid number the File Manager menu displays.

You can also select a program from the Debug menu.

7.1.4 Copying a Program

Function

Copy Program creates a new file and copies the selected program into it. This allows changing variables in the new program while preserving a copy of the original.

Operation

Press	Result
1. <FILE MGR>	display the File Manager menu
2. <F3>	display the Copy Program display and the prompt: COPY PROGRAM FROM PROG #: (program to be copied)
3. right <SHIFT> key twice	enable number entry
4. <NUMBER KEYS>	type in the number identifier of the program you want to copy
5. <⟵> left arrow	clear errors from the input field before you press <ENTER>
6. right <SHIFT> key once	disable number entry
7. <ENTER>	enter the number and select the program and display the prompt: COPY PROGRAM FROM PROG #: (program to be copied) TO PROG #: (file to contain the copy)
8. Repeat steps 3 through 7	Enter the number of the program to which the program is to be copied.
9. <FILE MGR>	display the File Manager menu
10. <F1>	Verify the copy in the Directory display.

7.1.5 Deleting a Program

Function

Delete Program removes unwanted MML programs to increase storage for new programs. Deleted programs no longer exist. Be sure you do not need the program or back up the program before you delete it.

Operation

Press	Result
1. <FILE MGR>	display the File Manager menu
2. <F4>	display the Delete Program display and the prompt: DELETE PROGRAM PROG #:
3. right <SHIFT> key twice	enable number entry
4. <NUMBER KEYS>	type in the number identifier of the program you want to delete
5. <⟷> left arrow	clear errors from the input field before you press <ENTER>
6. right <SHIFT> key once	disable number entry
7. <ENTER>	enter the number and display this prompt: DELETE PROGRAM PROG #: R U SURE <Y,N>:
8. left <SHIFT> and Answer the 'are you sure' prompt <Y> or <N>	
	When a valid number is entered and a <Y> or <N> key is pressed, the File Mgr main menu is displayed.
9. <FILE MGR>	display the File Manager menu
10. <F1>	Verify the copy in the Directory display.

7.1.6 Renaming a Program

Function

The Rename Program function changes the identification number of a program. You cannot change the source text code here. Only the program number identifying the program.

Programs are listed by name and number in the file management utilities in the Offline Development Station and in the IMC 120. Select programs are by number on the handheld pendant.

Use the program names only when you input the source text for the MML program.

You can find program names in the listing of program names and numbers found in the Directory display of the File Manager menu (See figure 7.1 on page 7-2).

When you change a number the IMC 120 directory listing updates.

Operation

Press	Result
1. <FILE MGR>	display the File Manager menu
2. <MORE>	display the More File Manager menu.
3. <F1>	display the Rename Program display and the prompt: RENAME PROGRAM OLD PROG #:
4. right <SHIFT> key twice	enable number entry
5. <NUMBER KEYS>	type in the number identifier of the program you want to delete
6. <←> left arrow	clear errors from the input field before you press <ENTER>
7. right <SHIFT> key once	disable number entry
8. <ENTER>	enter the number and display this prompt: RENAME PROGRAM OLD PROG #: NEW PROG #:
9. Repeat steps 3 through 7	Enter the new number of the program. When you enter both valid numbers, the Rename Program display returns.
10. <FILE MGR>	display the File Manager menu
11. <F1>	Verify the copy in the Directory display.

7.1.7 Displaying Program Information

Function

Program Information displays information about a program. When you enter a valid number at the PROG # prompt, the display containing the program number, program name, program size in bytes, and debug information status is displayed.

Operation

	Press	Result
1.	<FILE MGR>	display the File Manager menu
2.	<MORE>	display the More File Manager menu.
3.	<F2>	display the Program Information and the prompt: PROGRAM INFO PROG #:
4.	right <SHIFT> key twice	enable number entry
5.	<NUMBER KEYS>	type in the number identifier of the program you want to delete
6.	<←> left arrow	clear errors from the input field before you press <ENTER>
7.	right <SHIFT> key once	disable number entry
8.	<ENTER>	enter the number and return to the Program Information display.
9.	<FILE MGR>	display the File Manager menu
10.	<F1>	Verify the copy in the Directory display.

Debug Information status definitions:

Status Display	Definition:
OK -	Both the symbol names for the variables and the breakpoint step markers are available.
STEPS -	The symbol names for the variables have been removed. The breakpoint step markers are available.
SYMB -	The program was compiled without debug information. The symbol names for the variables are available. The breakpoint step markers were removed.
<blank> -	Both the symbol names for the variables and the breakpoint step markers are not available.

7.1.8 Removing Debug Information

After you test and debug a program, the variables information table may be cleared using the Remove Debug Info function from the handheld pendant.

Function
Remove Debug Information makes more MML program space available by removing the program symbols from memory. This frees up memory space for MML programs and also prevents anyone from changing the variables listed in the source program.

Remove Debug erases the debug symbols table that contains the Position, Integer, Boolean, and Real variables symbols for the active program. The variables remain intact, you delete only the symbol table accessing the variables.

Operation

	Press	Result
1.	< FILE MGR >	display the File Manager menu
2.	< MORE >	display the More File Manager menu.
3.	< F3 >	display the Remove Debug Information and the prompt: REMOVE DEBUG INFO PROG #:
4.	right < SHIFT > key twice	enable number entry
5.	< NUMBER KEYS >	type in the number identifier of the program you want to delete
6.	<←> left arrow	clear errors from the input field before you press < ENTER >
7.	right < SHIFT > key once	disable number entry
8.	< ENTER >	enter the number and delete the debug information. When you enter a valid number, the Remove Debug Info display is restored.
9.	< ENTER >	to exit this display.
10.	< MORE >	return to the first File Manager display.
11.	< FILE MGR >	display the File Manager menu
12.	< F1 >	Verify the copy in the Directory display.

8.0 Chapter Overview

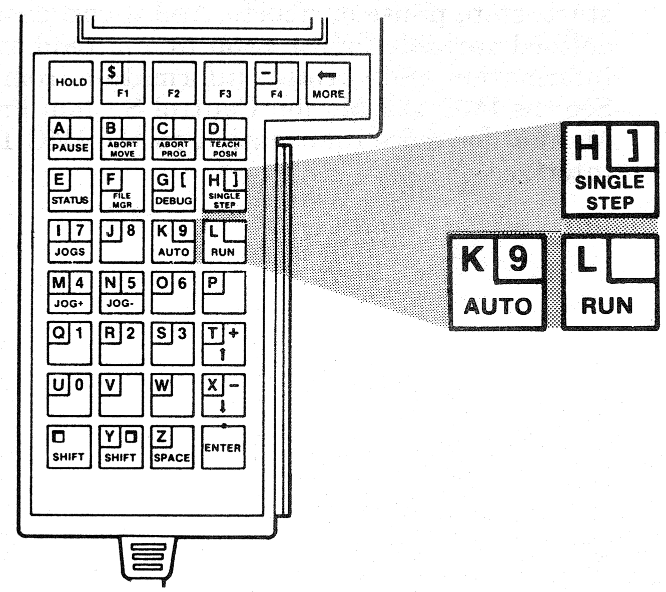
This chapter discusses how to run and monitor an MML program from the handheld pendant in auto mode. These procedures include:

- running a MML program
- stopping a MML program
- modifying the programmed speed

Figure 8.1. shows the keys to start and stop an MML program in auto mode.

Start a Program	Stop a Program
• Single Step key • Auto key • Run key	• Pause key • Hold key • Abort Move key • Abort Prog key

Figure 8.1
Keys for Automatic Operation



Monitor the IMC 120 status with the Status menu display keys:

- Jogs
- Debug
- Status
- File Manager

See section 2.5 for more information on these keys.

Important: Jogging operations (manual mode operations) either from the handheld pendant or from the PLC can only run in manual mode. MML programs can run only in auto mode.

Important: If you suspend an MML program and press the <JOGS> key to enter manual mode, all condition handlers and fast interrupt statements (FIN/FOUTs) associated with the program are disabled. You re-enable these condition handlers and fast interrupts when you press the <AUTO> key to return to auto mode.

8.1 Running a Program

Function

The PLC runs MML programs that have been downloaded and tested on the IMC 120. The PLC controls the IMC 120 with commands sent through the backplane interface.

The PLC provides system control information, program control information, and variable parameter information. The PLC can select which MML programs to start, stop, pause or abort. And it can pass user defined variable information to and read status information and variable information from the IMC 120. See the IMC 120 Motion Control System Programming Manual for more information on the IMC 120/PLC interface.

Operation

Run a program from the pendant with this procedure:

Press	Result
	select a program
1. <FILE MGR>	display the File Manager menu
2. <F2>	select a program.
at the PROG # prompt	
3. right <SHIFT> key twice	enable number entry
4. <NUMBER KEYS>	type in the number identifier of the program you want to run
5. right <SHIFT> key once	disable number entry
6. <ENTER>	enter the number and select the program
	select auto mode*
7. <AUTO>	put the servo controller module into automatic mode. You must be in automatic mode to run a program.
	start the program
8. <RUN>	start the program.
9. <SINGLE STEP>	key to change from continuous run to step mode, and vice versa.

Note:
* You can also select a program from the Debug menu.

You can run the program in either continuous or single step. The type of run you select is displayed in upper case letters on the Mode display of the Status menu.

8.1.1 Single Stepping a Program

Function

The <SINGLE STEP> key on the handheld pendant has a similar effect on the program and axis motion as pressing the the <PAUSE> key.

The <SINGLE STEP> key selects either single step or continuous operation for running an MML program. You can use either operation in the auto mode. Check if the MML program is running in single step or continuous on the Mode display of the Status menu.

When you press the <SINGLE STEP> key, a message displays momentarily on the display indicating single step is active. In the single step mode the MML program stops after each executable statement (program step) is completed. Press <RUN> to run the program through the next statement in the program.

Important: The asynchronous nature of the program interpreter and the motion planner is maintained while executing a program in single step mode. Axis movement and program execution can occur at the same time. In other words, if a motion statement with a NOWAIT clause is executed, it continues running even though the program interpreter has stopped, awaiting a <RUN> key.

A single step is an executable statement in the MML program. A single step may contain a single program statement or a statement with modifying statements.

The MML program listing has asterisks beside the line numbers where the execution stops while single stepping (if the program was compiled with the debug switch on). The asterisks indicate program steps.

See the IMC 120 Motion Control System Programming Manual for more programming information.

Operation

You can single step a program if you set the debug switch on, when compiling the program. Select the program to be run from the File Mgr or Debug menu.

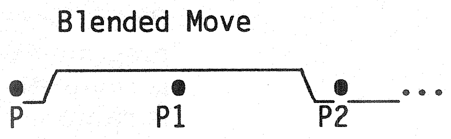
	Press	Result
1.	< SINGLE STEP > *	Select single step mode
2.	< RUN >	Proceed to the next step in the program.
3.	< STATUS > and < F2 >	monitor the MML program in step mode. The axis stops at each executable step in the MML program and the Status menu displays program and machine information.
4.	< RUN >	The axis proceeds to the next executable statement and displays the programmed motion.
5.	< ENTER >	return to the Status main menu

Note:
* Press the < SINGLE STEP > key again to select continuous operation again.

Figure 8.2
Example of a Single Step Statement

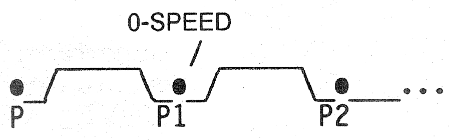
```
  •
  •
  •
41*    WITH TERMTYPE = NODECEL
42     MOVE TO P1 NOWAIT                 --  NOWAIT allows MOVE TO P2 to
43*    MOVE TO P2                        --  start when MOVE TO P1 begins.
44       'program execution'             --  NODECEL on line 41 allows
45                                       --  program execution to resume past
46                                       --  move to P2 when NODECEL is
47                                       --  satisfied.
  •
  •
  •
```

figure for statement -->
number 41 to 43



SINGLE STEPPING STATEMENT NUMBER 41 TO 43

figure for statement -->
number 41 to 43



•
•
•

8.2 Stopping Motion and Program

Function

Motion execution and program execution operate asynchronously. The program can be moving the axis, completing a move, and running other commands at the same time. Stopping the motion may not stop the program execution. See table 2.A in chapter 2.

Operation

The key you press to stop the motion or program (or both) is determined by how you want to stop the function, and how or if you want to start the function again. Table 2.A in chapter 2 compares the methods of stopping and starting an interrupted function (motion or program. Sections 8.2.1 through 8.2.3 illustrate stopping the axis motion and program together and separately.

8.2.1 Stopping and Resuming Motion and Program

Function

When you want to stop both motion and program, and restart them from the interrupt point, you need to press the <HOLD> key to stop the axis motion and the <PAUSE> key to stop the program.

Operation

Press the <HOLD> key on the handheld pendant to immediately decelerate the axis to to a controlled stop (jog moves and programmed MML motion). The IMC 120 software commands the axis to a stop through the analog velocity command to the servo drive. The drives remain on.

Press the <PAUSE> key on the handheld pendant to stop program execution after the statement that is running is complete. There is no effect on the statement that is running. Any motion that is in progress continues until finished. The <PAUSE> key has no effect on jog moves.

Important: The MOVE TO or MOVE BY statement eventually finishes, then execution stops. A MOVE_AT_SPEED must be cancelled by the program, operator, or PLC.

Press <RUN> to restart a program on pause or hold from the interrupt point.

	Press	Result
1.	< HOLD >	The axis motion stops. The display on the handheld pendant shows a message momentarily that the < HOLD > key is active.
2.	< PAUSE >	The program stops and no more motion is started.
3.	< RUN >	**Release hold and pause** The next executable program statement starts and axis motion begins if called for in the MML program.

You can also hold motion by setting the $SPEED__OVR system variable to zero.

8.2.2 Abort the Current Motion Only

Function

The <ABORT MOVE> key on the handheld pendant aborts the current motion of the axis. Program execution continues and the next motion command starts. The program execution environment compares programmed moves to travel and speed limits. If any move violates these limits because of the aborted move, an error is generated and the next move cannot start.

Operation

	Press	Result
1.	< ABORT MOVE >	aborts the current motion of the axis. The program execution continues.

8.2.3 Aborting Current Motion and Program

Function
Use the abort function to immediately stop the current motion and the program.

The <ABORT PROG> key on the handheld pendant stops program execution immediately. An aborted program resets to start from the beginning and waits for you to press <RUN>. An aborted jog or homing move is not resumable and must be restarted.

Pressing the <ABORT PROG> key immediately decelerates the axis to a controlled stop (jog moves and programmed motion). The IMC 120 software commands the axis to a stop through the analog velocity command to the servo drive. The drives remain on.

Operation

	Press	Result
1.	< ABORT PROG >	Stops program execution and axis decelerates to a controlled stop
2.	< RUN >	**Release from the pendant Abort state** start the program from the beginning.

8.3 Changing Speed

Changing the speed of the axis helps test axis motions and motion related functions in new programs. This list contains the ways to change axis speed:

- change the selected speed in the Jog Speed display
- change the $SPEED_OVR variable
- change the $SPEED variable

8.3.1 Changing Speed From the Jog Speed Display

Function
The speed values you entered into the AMP system parameters are displayed in the Jog Speed display. This display lets you select a specific speed to be active from the AMP speed values.

Operation
To change speed from the Jog Speed display, press the function key labelled next to the desired speed. See chapter 5 for information on setting jog speed.

8.3.2 Changing Speed Override

Function
The speed override value defines the speed of the axis when you command motion from an MML program or jog.

Setting the $SPEED__OVR system variable to zero affects moves in progress and any moves in the queue. The amount of override also affects moves started after the $SPEED__OVR is changed. The override value may be from 0 to 127 in increments of 1.

Operation
Enter the value to the left of the equal sign (into the variable) to get the percent override listed on the right of the equal sign. These values are examples only.

```
127 = 127% of jog or programmed speed
100 = 100% of jog or programmed speed
 50 = 50% of jog or programmed speed
 20 = 20% of jog or programmed speed
  0 = 0% of jog or programmed speed
```

Note that jog includes the home function.

8.3.3 Changing $SPEED Variable

Function
The $SPEED system variable determines axis speed. The axis speed changes to the value in the $SPEED system variable.

The $SPEED system variable cannot affect moves in progress. Only moves started after it is changed are affected.

Operation
Enter the value of the speed you want for the axis into the $SPEED system variable. The value you enter must be between the maximum and minimum speed allowed. See the Modifying Variables section of Chapter 6.

A.0 Appendix Overview

This Appendix contains error messages displayed on the handheld pendant or sent to the PLC for handling.

A.1 Messages Displayed On The Handheld Pendant

Table A.A beginning lists error messages that are displayed on the handheld pendant and the causes of the messages and ways to correct them.

Table A.A
Error Messages

ERROR#	ERROR MSG	CAUSE(S)	RECOVERY STEPS
768	ABORTING STATEMENT	In the MML application program the ABORT statement got executed. Just an informatory message.	The program can be run again. First determine why the ABORT statement got executed. Avoid any bad operation that caused the execution of the ABORT statement.
906	A CORRUPT FILE WAS DELETED	1. Control powered down in the midst of a sensitive file move routine	1. Redownload the program
1027	ACCELERATION LIMIT *ERROR*	The programmed accdec ($ACCDEC) does not fall within the legal limits; 0 < accdec < = 1.0; remember that $ACCDEC is a percentage of MAX_ACCEL (an AMP value).	Programming error; program $ACCDEC to be greater than 0, but less than or equal to 1.0.
129	ALREADY PAUSED *WARN*	1. Program is already paused	1. Resume the program, clear the value

ERROR#	ERROR MSG	CAUSE(S)	RECOVERY STEPS
2440	ARRAY ASSIGNMENT ERROR	**NOTE** Not supported at run time in IMC 120 at present.	
2441	ARRAY BOUNDARY EXCEEDED *ERROR*	1. Attempted to access a variable outside of the array	1. Rewrite the MML program so as not access variables outside of the array this will cause the program to abort
138	AXIS ALREADY IN MOTION *WARN*	1. Axis is in motion	1. Try again once axis motion ceases
2434	BAD DIGITAL I/O LINE NUMBER	The number specified for din or dout line is either less than one or more than the maximum supported in the IMC 120 system.	Correct MML application program logic.
2438	BAD FAST I/O INDEX	The number specified for fin or fout line is either less than one or more than the maximum supported in the IMC 120 system.	Correct the MML application program logic.
2439	BAD COND. HNDLR. INDEX	The number specified for condition handler is either less than one or more than the maximum supported in the IMC 120 system.	Correct the MML application program logic.
2433	BAD TIME VALUE	The time specified in a MML DELAY statement is negative or greater than 24 hours (86,400 seconds).	Correct MML application program logic.
920	CAN'T ARM PROBE WHEN NO PROBE	1. $PROBE__ARMED can't be modified to TRUE via the handheld pendant when there is no probe or linear probe.	1. Hit MORE key on HHP and select another variable to modify or change probe type to be either trigger probe AMP utility

ERROR#	ERROR MSG	CAUSE(S)	RECOVERY STEPS
919	CAN'T CHANGE WHEN NOT RATIOING	1. $DELPHASE can't be modified to a value other than the current value when not RATIOing.	1. Hit MORE key on HHP and select another variable to modify or RATIO the axis to a master via a MML program and try to modify $DELPHASE again
917	CANT RUN AXIS NOT HOMED	1. Attempt to run a program before the axis has been homed after power up.	1. Home the axis. Try to run the program again.
915	CANT RUN CONTROL IN ESTOP	1. Attempt to start or resume a program while in estop.	1. Perform an Estop Reset operation, then try to run the program again.
916	CANT RUN TEACHING	1. Attempt to run a program while the control is "waiting" for a position or variable to be taught.	1. Abort the program and attempt to run it again OR teach the position or variable.
2560	CANNOT ARM FAST I/O	The MML application motion clause ARM is specifying a fin or fout that is not yet posted.	Correct the MML application program logic.
2688	CANNOT ENABLE UNPOSTED COND. HNDLR.	The condition number specified in a MML ENABLE statement is not yet posted.	Correct the MML application program logic
130	CANNOT JOG IN AUTO MODE *WARN*	1. Jog request made with control in auto mode	1. Put control in Manual mode and resubmit the request
131	CANNOT JOG IN AUTO MODE *WARN*	1. Jog request made with control in auto mode	1. Put control in Manual mode and resubmit the request
3584	CANNOT SERVICE FAST IN *ERROR*	Too many fast input interrupts and the finfout task is not able to respond fast enough or the finfout task is not created when no program is running	ESTOP ERROR; run a MML application program

ERROR#	ERROR MSG	CAUSE(S)	RECOVERY STEPS
2	CANNOT SWITCH MODES *WARN*	Operator attempted to change modes from auto to manual by pressing the jogs key. Unfortunately, either the motion is not complete or a program is running. Given that one of these conditions is true, the operator can't change mode.	Operator must wait for motion to complete and the program status ($PGM__STATUS) is either not running or suspended. Then he can change modes with the jogs key.
4	CANNOT RESET AMP NOT LOADED *WARN*	Tried to exit estop before AMP has been loaded, or after an unsuccessful AMP download.	Successfully download AMP.
3	CANNOT RESET NON-RECOVERABLE *WARN*	Tried to exit estop after a non-recoverable estop has occurred.	Power down then back up (currently, there are no non-recoverable errors present in the systems.
128	CANNOT RESUME *WARN*	1. No pause currently on control	1. Clear the value
139	CANNOT RETURN IN AUTO MODE *WARN*	1. Control is in auto mode 2. Homing operation has occurred since manual mode was entered 3. Homing or jogging in progress	1. Place control in manual, reprogram 2. None 3. None
908	CONVERSION ERROR	1. Array index is not numeric when try to modify array 2. Value to modify an integer or a real variable is not entered as a numeric value 3. Value to modify a boolean variable is not entered as 1, 0, TRUE, or FALSE 4. Wrong data type has been detected or illegal value.	1. & 2. Hit MORE key on HHP and enter a numeric value 3. Hit MORE key on HHP and enter value with 1, 0, TRUE, or FALSE 4. Enter correct data type or value.

ERROR#	ERROR MSG	CAUSE(S)	RECOVERY STEPS
2051	DIVIDE BY ZERO	Divide by zero attempted.	Correct MML application program logic.
3080	EXCESS FOLLOWING ERROR *ERROR*	The value of the axis following error exceeded the limit established in AMP. This can be caused by an obstruction to axis motion, servo wiring error, or loss of feedback channel. This error causes estop.	Reset from estop. If the problem persists, check axis mechanics, wiring, and feedback device.
2561	FAST I/O CODE *ERROR*	Bad codes in finfout action	Abort the program; run a "good" MML application program
3712	FAST OUTPUTS DISABLED *HARDWARE ERROR*	A fast output is shorted. This disables all fast outputs and causes estop.	Clear the output fault then reset from estop.
513	FEEDBACK IS MARGINAL (MASTER) *WARN*	IMC 120 hardware detected a feedback failure on master feedback device. This error is not fatal unless it occurrs multiple times during a 9.6ms iteration.	None; this is a warning message only. However if this message appears often, the feedback device should be checked.
512	FEEDBACK IS MARGINAL (SLAVE) *WARN*	IMC 120 hardware detected a feedback failure on the slave feedback device. This error is not fatal unless it occurs multiple times during a 9.6ms iteration.	None; this is a warning message only. However if this message appears often, the feedback device should be checked.
896	FILE COPY ERROR	1. Destination file already exists 2. Control out of memory	1. Rename the destination 2. Make room in memory

ERROR#	ERROR MSG	CAUSE(S)	RECOVERY STEPS
897	FILE DELETE ERROR	1. File does not exist 2. File already open	1. See if it exists or not 2. If selected, abort the program; otherwise, stop any file operations on the file.
899	FILE OPEN *ERROR*	1. File could be locked 2. Too many open files on system 3. Attempt to read non-existent file 4. Attempt to write to a file currently opened for read 5. Attempt to write to a file currently opened for write	1. Try again, file was being moved 2. 6 max, close one and try again 3. Make sure the file exists 4. Close file and try again 5. Close file and try again
2193	FILE OPEN FAILED	This is a system error	Report to Allen-Bradley
898	FILE RENAME ERROR	1. Source or destination file doesn't exist 2. Destination file already exists	1. Check the file names 2. Rename the destination
903	FILE WRITE ERROR	1. Out of ram on control 2. Attempt to write to a nonexistent file 3. Attempt to write to an unopened file 4. Attempt to write to a file opened for read	1. Delete a file, try again 2. Create the file 3. Open the file 4. Close the file, open for write
2690	GLOBAL COND. HNDLR. CODE *ERROR*	Bad codes in global condition handler action	Abort the program; run a "good" MML application program
3456	HARDWARE ESTOP	The user estop string has opened.	Insure that the device which opened the string is functioning properly, then reset from estop.

ERROR#	ERROR MSG	CAUSE(S)	RECOVERY STEPS
256	HOME OPERATION ABORTED *WARN*	(1) released JOG+ or JOG-button (2) broken or incorrectly wired home switch	(1) n/a (2) replace or correctly wire home switch.
263	HOMED SUCCESSFULLY	1. Informational message saying the homing operation requested by the PLC or HHP was performed successfully.	1. N/A
258	HOMED TOO FAST *WARN*	Axis ran into the switch at too high velocity.	Slow down the speed at which the axis runs into the home switch.
909	ILLEGAL ARRAY LENGTH	1. In debug or teach mode, a variable which is not defined as an array is entered with [index] 2. In debug or teach mode, a variable is entered with [index] followed by some character other than ENTER key	1. Hit MORE key on HHP and enter the variable which is not defined as an array without [index] 2. Hit MORE key on HHP and enter the variable (if not an array, without [index]) (if an array, type [index] after the variable name)
2436	ILLEGAL EVENT NUMBER	The event number specified in a SIGNAL statement or action or EVENT condition is not in the range of 1-1000.	Correct the MML application program logic.
2437	ILLEGAL ERROR NUMBER	An illegal number was specified in an ERROR condition. Error numbers must be in range 1-64999.	Correct the MML application program logic.
133	ILLEGAL JOG INCREMENT *WARN*	1. Value < 0 or > 6	1. Reprogram the increment value
134	ILLEGAL JOG SPEED *WARN*	1. Value < 0 or > 3	1. Reprogram the speed value

ERROR#	ERROR MSG	CAUSE(S)	RECOVERY STEPS
135	ILLEGAL JOG TYPE *WARN*	1. Value < 0 or > 3	1. Reprogram the jog select value
910	ILLEGAL NAME LENGTH	1. In debug or teach mode, the name length of the variable to change is more than 12	1. Hit MORE key on HHP and enter the variable to change with right length
907	INCORRECT ARRAY SIZE	1. In debug or teach mode, array size is out of range	1. Hit MORE key on HHP and enter this array with right range
2050	INTEGER OVERFLOW	A MML application program operator produced a result greater than the maximum permissible integer value of 0000000H.	Correct MML application program logic.
902	INTERNAL ERROR CONTACT SERVICE	(1) Could not find uninitialized variable within a program symbol table. (2) Pendant tables corrupt causing illegal menu or item selection.	(This error is not fatal; the variable just won't be taught). Unplug pendant and reinsert (this is fatal to the serial task; other tasks are not affected).
140	INVALID PLC BLOCK ID *WARN*	1. Block no < 0 or > 7	1. Check the value, reprogram it
1028	JERK LIMIT *ERROR*	The programmed jerk ($JERK) does not fall within the legal limits; 0 < jerk < = 1.0; remember $JERK is a percentage of MAX_JERK (an AMP value)..	Programming error; program $JERK to be greater than 0, but less than or equal to 100.
257	JOG AXIS BEFORE HOMING *WARN*	System has not seen a NULL marker since power-up.	Jog the axis in either direction at least one revolution of the feedback device; then home the axis.

ERROR#	ERROR MSG	CAUSE(S)	RECOVERY STEPS
2689	LOCAL COND. HNDLR. CODE *ERROR*"	Bad codes in local condition handler action	Abort the program; run a "good" MML application program
3200	LOST PLC COMMUNICATIONS *ERROR*	No communication with the PLC has occurred within a reasonable period of time (approximately 5 seconds). This error causes estop.	Reset from estop. If the problem occurs again, check the PLC.
3079	MASTER FEEDBACK FAILURE *ERROR*	Multiple feedback failures were detected on the master feedback device during a 9.6ms iteration.	Check the master feedback device, then reset from estop.
3073	MASTER FEEDBACK HIGH *ERROR*	Feedback value from the master device was higher than physically possible for the device used. Caused by wrong no. of feedback counts: may be miswired, wrong AMP settings, or noise. This error causes estop.	Reset from estop. If the problem persists, check check the wiring for noise, accurate wiring. Also, check AMP for gated or non-gated marker and AMP set for correct feedback counts.
3075	MASTER-FEEDBACK LOW *ERROR*	Feedback value from the master device was lower than physically possible for the device used. Caused by wrong no. of feedback counts: may be miswired, wrong AMP settings, or noise. This error causes estop.	Reset from estop. If the problem persists, check check the wiring for noise, accurate wiring. Also, check AMP for gated or non-gated marker and AMP set for correct feedback counts.
1152	MASTER POSITION TOLERANCE *ERROR*	The master position feedback speed is greater than the AMP specified master position feedback tolerance.	Programming error; slow the master axis prior to entering RATIO mode to within the AMP specified tolerance or increase the AMP value.

ERROR#	ERROR MSG	CAUSE(S)	RECOVERY STEPS
3077	MASTER QUADRATURE FAULT *ERROR*	The IMC 120 detected simultaneous transitions on the A and B channels of the master encoder. This is an illegal condition. This error causes estop.	Reset from estop. If the problem persists, check the wiring to remove sources of noise. Also check the encoder, it may have failed.
640	MEMORY CARD BATTERY LOW *WARN*	Battery backing up memory card is low	Upload all files and AMP to ODS; replace battery; download all files and AMP to the IMC 120
2182	COND. HNDLR. NESTING EXCEEDED	The maximum number of nesting allowed in calling condition handler routines has been exceeded.	Correct the MML application program logic.
2177	MOTION QUEUE OVERFLOW *ERROR*	Too many programmed motions are being attempted; happens as result of many STOPped motions being RESUMEd all at once.	Programming error; the program must be modified, recompiled, redownloaded, and and re-executed.
2179	MOTION STACK OVERFLOW *ERROR*	Too many STOPped motions without any of them being RESUMEd.	Programming error; the program must be modified, recompiled, re-downloaded, and re-executed.
260	MOVE TO ALT HOME ABORTED *WARN*	1. The Move To $ALT__HOME has been aborted by the PLC.	1. Start the function from the PLC again
262	MOVE TO ALT HOME SUCCESSFUL	1. Informational message saying the Move To $ALT__HOME requested by the PLC was performed successfully.	1. N/A
2052	NEGATIVE DIVISOR	The operand on the right of a MOD operator was negative. The operand must be greater than zero.	Correct MML application program logic.

ERROR#	ERROR MSG	CAUSE(S)	RECOVERY STEPS
905	NO DEBUG INFORMATION *ERROR*	(1) Tried to enter teach and no program symbols were found for the active program. Even though program is selected, symbol information is not valid until it begins to execute.	Press MORE key (must do on any error select a program and execute.
		(2)Tried to set a break point and file header was set to DEBUG__FLAG equals false, meaning the program was compiled without debug option specified.	Press MORE key. Recompile with debug switch.
914	NO PROGRAM ACTIVE	1. In debug mode, a variable to change is not found in system symbol table, nor in special symbol group and no program is selected	1. Select the desired program and hit RUN key and then in debug mode enter the variable which is defined in the selected and active program.
1	NO PROGRAM SELECTED *WARN*	RUN is pressed without ever selecting a program	Select a program that is already downloaded in the control
1024	OVERTRAVEL *ERROR*	The programmed position has exceeded an AMP specified positive or negative overtravel.	Programming error; program position somewhere within the overtravel limits or extend the limits.
384	OVERTRAVEL *WARN*	Jog or speed move encounters a software overtravel	Push the "more" key
2178	PLANNED MOTION QUEUE OVERFLOW *ERROR*	Too many motions have been planned.	Internal software error; if this happens there's nothing a user can do to fix it; there is a system software problem.

ERROR#	ERROR MSG	CAUSE(S)	RECOVERY STEPS
137	PLC BCD VALUE INVALID *WARN*	1. BCD value transferred is larger than the IMC-120 can handle.	1. Check the value, reprogram it
1408	PLC IORESET REQUEST	The PLC has gone into test or program mode; this causes a motion pause.	Press resume.
2184	PLNG EVENT QUE OVERFLOW *ERROR*	1. Sytem Error	1. Report to AB
771	PAUSE STATEMENT DONE	In the MML application program the PAUSE statement got executed. Just an informatory message.	The program can be continued just by pressing RUN.
1025	POSITION LIMIT *ERROR*	The programmed position is too large to fit into an internal position variable.	Programming error; position calculation in the MML application program must be restrained to physical limits.
904	PROGRAM ACTIVE *WARN*	Tried to select a program and a previously selected program is still active.	Press the MORE key. Press abort program. Now try selecting program.
2183	PROGRAM EVENT Q OVERFLOW *ERROR*	1. Sytem Error	1. Report to AB
2180	PROGRAM MOTION QUEUE OVERFLOW *ERROR*	The program motion queue is full when attempting a jog while the program is suspended.	The program will have to be ABORTed in order to clear the programmed motion queue; this error should actually never happen; it is only being checked as a precaution.
900	PROGRAM NOT FOUND	Any file management function that requires a source file will search the directory to verify the source file entered exists.	Press MORE key. Choose a file that is in the directory.

ERROR#	ERROR MSG	CAUSE(S)	RECOVERY STEPS
2304	RATIO OVERTRAVEL *ERROR*	If there are overtravel limits present OR the axis has not been homed AND the program attempts to begin RATIOing this error occurs.	Programming error: overtravels may not be present on an axis that is RATIOing to a master; also, the axis must have been homed prior to attempting the RATIO.
769	REACHED END OF PROGRAM	Just an informatory message. Either an END statement or a RETURN statement was executed in the MML main program.	The program can be run again just by pressing RUN.
770	REACHED BREAKPOINT AT STEPNO	A break point programmed has been reached.	The program can be continued just by pressing RUN.
912	READ ONLY VARIABLE	1. In debug mode, a value is entered to change a variable which is a read only variable in the system variable table or is DIN or is FIN	1. Hit MORE key on HHP and enter this variable name again, after the value is shown on HHP, just hit ENTER key again
2048	REAL OVERFLOW	A MML application program operation produced a result greater than the maximum permissable real value.	Correct MML application program logic.
2049	REAL UNDERFLOW	A MML application program operation produced a result less than the minimum permissible real value.	Correct MML application program logic.
259	RETURN TO POSN ABORTED *WARN*	1. The Return To Posn has been aborted by the PLC or 2. by releasing the [JOG +] or [JOG-] key on the HHP.	1. Start the function from the PLC again 2. Start the function from the HHP again by pressing either the [JOG +] or [JOG-] key

ERROR#	ERROR MSG	CAUSE(S)	RECOVERY STEPS
261	RETURN TO POSN SUCCESSFUL	1. Informational message saying the Return To Posn requested by the PLC or HHP was performed successfully.	1. N/A
1281	ROUND OR TRUNC. OUT OF RANGE	The real value supplied as an argument to a round or truncate function was greater than the maximum permissible integer in the MML application program.	Correct MML application program logic.
2176	RUN TIME STACK OVERFLOW	The MML stack has overflowed. The reasons are (1) The subroutine in the MML application program is calling itself either directly or indirectly too many times. (2) The subroutine calls are nested too deep.	In case one (1) correct program logic. In the second (2) case try to reduce number of variables, parameters or nesting.
3078	SLAVE FEEDBACK FAILURE *ERROR*	Multiple feedback failures were detected on the slave feedback device during a 9.6ms iteration.	Check the slave feedback device, then reset from estop.
3072	SLAVE FEEDBACK HIGH *ERROR*	Feedback from slave device was a value higher than physically possible for the device used. Caused by wrong no. of feedback counts: may be miswired, wrong AMP settings, or noise. This error causes estop.	Reset from estop. If the problem persists, check the wiring for noise, accurate wiring. Also, check AMP for gated or non-gated marker and AMP set for correct feedback counts.

ERROR#	ERROR MSG	CAUSE(S)	RECOVERY STEPS
3074	SLAVE FEEDBACK LOW *ERROR*	Feedback from the slave device was a value lower than physically possible for the device used. Caused by wrong no. of feedback counts: may be miswired, wrong AMP settings, or noise. This error causes estop.	Reset from estop. If the problem persists, check the wiring for noise, accurate wiring. Also check AMP for gated or non-gated marker and AMP set for correct feedback counts.
3076	SLAVE QUADRATURE FAULT *ERROR*	The IMC 120 detected simultaneous transitions on the A and B channels of the slave encoder. This is an illegal condition. This error causes estop.	Reset from estop. If the problem persists, check the wiring to remove sources of noise. Also check the encoder, it may have failed.
918	SORRY YOUR PASSWORD IS INCORRECT	1. Password was incorrect	1. Try again
1026	SPEED LIMIT *ERROR*	The programmed speed ($SPEED) does not fall within the legal limits; 0 < speed < = MAX__SPEED, where MAX__SPEED is an AMP value	Programming error; program $SPEED to be greater than 0, but less than or equal to MAX__SPEED.
1280	SQUARE ROOT OF NEGATIVE NUMBER	The square root of negative number is being taken.	Correct MML application program logic.
2185	STMT EVENT Q__0 OVERFLOW *ERROR*	1. Sytem Error	1. Report to AB
2186	STMT EVENT Q__1 OVERFLOW *ERROR*	1. Sytem Error	1. Report to AB
2187	STMT EVENT Q__2 OVERFLOW *ERROR*	1. Sytem Error	1. Report to AB
1288	STMT EVENT Q__3 OVERFLOW *ERROR*	1. Sytem Error	1. Report to AB

ERROR#	ERROR MSG	CAUSE(S)	RECOVERY STEPS
2189	STMT EVENT Q__4 OVERFLOW *ERROR*	1. Sytem Error	1. Report to AB
2190	STMT EVENT Q__5 OVERFLOW *ERROR*	1. Sytem Error	1. Report to AB
901	STEP NUMBER NOT FOUND	tried to set a break point and a break point could not be found as an executable statement in the current program.	Press the MORE key. Enter step number of executable statement.
2435	SUBSCRIPT RANGE	The subscript of an array variable is either less than one or greater than the size of the array.	Check MML application program logic to determine how the illegal subscript value was generated.
1029	TERMTYPE OUT OF RANGE *ERROR*	$TERMTYPE has been given a value other than an enumerated value forNODECEL, NOSETTLE, COARSE, and FINE.	Programming error; $TERMTYPE enumerated value must represent either NODECEL, NOSETTLE, COARSE, or FINE.
2181	TOO MANY EVENTS QUEUED	A cascade of events occured. A maximum of ten events can be queued.	Correct the MML application program logic.
2191	TOO MANY FAST I/O *ERROR*	1. More than 10 fout actions executed	1. Correct MML program
2192	TOO MANY COND. HNDLR. ROUTINES *ERROR*	1. More than 10 monitor routines called in condition handler(s)	1. Correct MML program
2691	UNINIT VAR IN MON OR FASTIN	Uninitialized variables have been used to monitor FIN/FOUT conditions or actions. This is illegal. The program aborts.	Either teach the variable or correct the MML program. Press the run key to start the program over from the beginning.

ERROR#	ERROR MSG	CAUSE(S)	RECOVERY STEPS
2432	UNINITIALIZED LOCAL VARIABL	A variable defined in a routine is being used before assigning any value to it.	Correct the MML application program logic.
1030	UNINITIALIZED STATIC VARIABLE	A static variable was not initialized before being used	Teach the variable or abort program
3328	USER POWER LOST	The user power supply has been shorted or or disconnected. This error causes estop.	Check the user power and its connections to the PLC. After it is fixed, reset from estop.
913	VALUE TOO LARGE/SMALL FOR VARIABLE	In debug or teach mode, the value entered to change a variable is out of its range	1. Hit MORE key in HHP and enter this variable with a value within the range
911	VARIABLE NAME NOT FOUND	In debug or teach mode, the variable to change is not found in the MML program variable list, or system variable table, or special variable group	1. Hit MORE key on HHP and enter another variable which is defined; an MML program variable, or system variable, or special variable (FIN, etc.)

B.0 Appendix Overview

This Appendix contains System Parameters and System Variables accessable from the handheld pendant.

B.1 System Parameters Accessable From The Handheld Pendant

The following list contains the System (AMP) Parameters that are accessable from the handheld pendant. See the IMC 120 Motion Control System Integration Manual for more information on integrating the IMC 120 System.

B.1.1 Changing Parameter Values

Each system parameter that can be altered from the handheld pendant has two names. One parameter name appears on the handheld pendant display and the other parameter name is found on the ODS screen.

The following parameters are listed as they appear on the handheld pendant. The parameter name in brackets is the name that appears on the ODS screen when you use the ODS to change values.

The number following the parameter name is he parameter number you call up from the handheld pendant or the ODS to change the parameter value.

REV MAIN FB (Invert Feedback Signals) No. 3070

USAGE: Enables or disables inverting of the signal input to the controller from the feedback device.

RANGE: Yes (feedback signals inverted) or No (feedback signals not inverted)

DEFAULT VALUE: No

NOTES: Inverting the position feedback signal effectively changes the direction of axis motion indicated by the feedback. Consequently, this parameter provides an easy method of correcting for position feedback inputs which have been wired so that the axis moves in the direction commanded by the DAC output, but the axis position register indicates the axis is moving in the opposite direction. (Compare Invert D to A Output).

INVERT D/A (Invert D/A Output) No. 2060

USAGE: Enables or disables inversion of the signal from the controller to the axis drive.

RANGE: Yes (D/A output inverted) or No (D/A output not inverted)

DEFAULT VALUE: No

NOTES: This parameter provides an easy way to correct for D/A outputs which have been wired so that the axis to move in the opposite direction from the commanded direction.

INVERT PROBE (Polarity of Probe) No. 3080

USAGE: Specifies the type of contact used on the touch probe.

RANGE: Normally open or Normally closed.

DEFAULT VALUE: Normally open

NOTES: None

REV AUX FB (Invert 2nd Axis Feedback) No. 3071

USAGE: Enables or disables inverting of the signal input to the controller from the master feedback device.

RANGE: Yes (feedback signals inverted) or No (feedback signals not inverted).

DEFAULT VALUE: No

NOTES: Inverting the position feedback signal effectively changes the direction of axis motion indicated by the feedback. Consequently, this parameter provides an easy method of correcting for channel 2 position feedback inputs, where the master axis moves in the opposite direction from the intended direction, but channel 1 position feedback inputs, slave feedback, and DAC output are correct.

B.1.2 System Variables Changed From the Handheld Pendant

This section describes the system variables that you can access from the handheld pendant. It gives the description and method of access for each system variable.

access system variables from the debug menu by typing in the system variable name at the name prompt in the variables display of the debug menu.

Type in the name of the variable directly into the handheld pendant to access the variable. For example, to read the value of $ERROR, type in $ERROR into the debug menu. The value currently in the $ERROR system variable is displayed.

The following variables can be accessed by the handheld pendant through the debug menu. See Chapter 6 in this manual for the procedure to modify variables.

Direct Read/Direct Write System Variables

$ACCDEC	$JERK	$RATIO_TYPE
$AT_POSN_TOL	$LPLCO[n]	$SPEED
$BPLCO[n]	$OFFSET	$SPEED_OVR
$DISABLE_OVR	$PHASE	$SPLCO
$DISABLE_PLC	$RATIO_OVR	$TERMTYPE

Direct Read/Built-in Write System Variables

$ALT_HOME	ALT_HOME built-in function
$DELPHASE	DELPHASE built-in function
$GAIN	GAIN built-in function
$PROBE_ARMED	ARM_PROBE and DISARM_PROBE built-in procedures
$RESULT	RESULT condition handler action
$UNITS	UNITS built-in function

Built-in Read/Built-in Write System Variables

$RATIO	GET_RATIO built-in procedure -read SET_RATIO built-in function - write

$ACCDEC -- Percentage of Maximum Acceleration/Deceleration

Use:

Determines the rate of acceleration and deceleration used for the next axis move, as a percentage of the Maximum Accel/Decel Ramp (system parameter 1110)

Range:

```
0.0 < $ACCDEC <= 100.0
```

```
where 0.0 is 0%, 100.0 is 100% of the Maximum
Accel/Decel Ramp
```

Notes:

- If the range is exceeded, a run-time motion planning error will result.

Data Type: real

MML Read Access: direct

MML Write Access: direct

HHP Read Access?: yes

HHP Write Access?: yes

$ALT__HOME -- Alternate Home (Retract) Position

Use:

This variable contains the position the axis will move to when an ALT HOME operation is commanded from the PLC.

Range:

```
Negative Overtravel Limit < $ALT_HOME < Positive Overtravel Limit
```

(if Software Overtravels Used? is Yes)

Notes:

- $ALT__HOME must be programmed within the axis overtravel limits (if used) or an error value of false will be returned, and the last value stored in $ALT__HOME will not be changed. Otherwise, a value of true will be returned and $ALT__HOME will be updated.
- The control must have been homed before you do an ALT HOME from the PLC.
- Unless some action has been taken to change it, $ALT__HOME is the same as machine home. After the axis has been homed, $ALT__HOME may be changed in 3 ways: 1) with the handheld pendant, 2) ALT__HOME built-in, 3) initialize home operation from the PLC.

There are 3 home positions:

- machine home -- the dimension set with AMP, representing a position along the axis to be home
- alternate home -- any dimension, a position relative to machine home, but not necessarily machine home. Alternate home is <u>never</u> outside the software travel limits.
- initialize home -- set by the PLC in block 0, word 3, bit 5. This lets you define a new zero point while keeping the original stored.

Data Type: position
MML Read Access: direct
MML Write Access: ALT__HOME(x) built-in function

```
status = ALT_HOME(alt_pos)
```

where alt__pos is a position expression that specifies the alternate home position desired.
HHP Read Access?: yes
HHP Write Access?: yes

$AT__POSN__TOL -- Position Tolerance for AT POSITION Condition

Use:

Specify the tolerance band surrounding a position specified in the condition portion of a condition handler.

Range:

```
-real limit < $AT_POSN_TOL < +real limit
```

Notes:

- Consider the following example for how $AT_POSN_TOL works:

  ```
  $AT_POSN_TOL = real_ exp

  MOVE TO p2,
    WHEN AT POSITION p1 DO
     actions
  ENDMOVE
  ```

  The $AT__POSN__TOL value specifies a band around p1 as follows:

```
       -$AT_POSN_TOL     +$AT_POSN_TOL
                   |         |
                   V         V
                   |    O    |              O
--------->_________|_________|______________
                   |    P1   |              P2
moving towards
P2 through P1
```

Data Type: real

MML Read Access: direct

MML Write Access: direct

HHP Read Access?: yes

HHP Write Access?: yes

$BPLCO -- Array of 112 Boolean Outputs

Use:

This array provides 112 boolean outputs from the IMC 120 to the host PLC via block transfers.

Range:

```
$BPLCO[n] = true -- element n is ON
$BPLCO[n] = false -- element n is OFF
```

Notes:

- Elements of $BPLCO are found in the following PLC blocks:

$BPLCO[n]	PLC block number
$BPLCO[1 to 16]	1
$BPLCO[17 to 32]	2
$BPLCO[33 to 48]	3
$BPLCO[49 to 64]	4
$BPLCO[65 to 80]	5
$BPLCO[81 to 96]	6
$BPLCO[97 to 112]	7

- You can use $BPLCO elements as general purpose boolean variables in an MML application program if they are not used for communicating information to the PLC. This provides a practical way to conserve memory (i.e., user RAM) since $BPLCO elements use system RAM.

Data Type: boolean

MML Read Access: direct

MML Write Access: direct

HHP Read Access: yes

HHP Write Access: yes

$DELPHASE -- Adjust Master/Slave Phase Relationship in Ratioing

Use:

This variable is used to offset phase errors that are introduced between the master and slave axis due to following error and time lag effects while ratioing.

Range:

```
-360.0 < $DELPHASE < 360.0
```

Notes:

- The units for $DELPHASE are in electrical degrees of the feedback device. For example, if a resolver is used (4000 counts per electrical revolution), a $DELPHASE value of 90 will produce an offset of 1000 resolver counts (1/4 electrical turn).

- Note that $DELPHASE is only active while ratioing is active. Also note that this offset is not interpolated into the axis motion. Rather, it is all dumped in at once. Therefore, it is the programmer's responsibility to change $DELPHASE in small increments, or live with (potentially) large amounts of axis jerk.

Data Type: real

MML Read Access: direct

MML Write Access: status = DELPHASE(x) built-in function, where x is a real expression for the number degrees to adjust the phase relationship

HHP Read Access?: yes

HHP Write Access?: yes

$DISABLE__OVR -- Disable Motion Speed Override

Use:

This variable determines whether or not the $SPEED__OVR variable will be applied to axis motion

Range:

```
$DISABLE_OVR = true -- $SPEED_OVR disabled
$DISABLE_OVR = false -- $SPEED_OVR enabled
```

Note:

- Note that any change to $DISABLE__OVR made during a move does not take effect until the next move.

Data Type: boolean

MML Read Access: direct

MML Write Access: direct

HHP Read Access?: yes

HHP Write Access?: yes

$DISABLE__PLC - Disable Communications with the PLC

Use:

This variable determines whether certain signals/functions from the PLC should be ignored or not. The PLC signals affected by the state of this variable are:

Single Transfer Operations	Block Transfer (Block 0) Operations
resume	dry run select
pause	jog increment select
override enable	jog speed select
alt home	jog type select
abort	initialize home
auto/manual	return to position
jog +	speed override value
jog -	speed override select
single step	program select
	program number
	offset select
	offset value
	preset select
	preset value

Range:

```
$DISABLE_PLC = true -- above signals/functions disabled
$DISABLE_PLC = false -- above signals/functions enabled
```

Notes:

Data Type: boolean

MML Read Access: direct

MML Write Access: direct

HHP Read Access?: yes

HHP Write Access?: yes

$GAIN - Position Loop Gain

Use:

The value of this variable determines the position loop gain as a percentage of the Maximum Axis Gain Value specified in AMP.

Range:

```
0.0 < $GAIN <= 1.0
```

```
where:
```

```
0.0 corresponds to 0%, and 1.0 corresponds to 100% of
the Maximum Axis Gain Value in AMP (system parameter
2620)
```

Notes:

- If the value passed to the GAIN built-in function is outside the range $0 < x <= 1.0$, a status of false will be returned and $GAIN will be unchanged. Otherwise a status of true will be returned and $GAIN will be changed.

Data Type: real

MML Read Access: direct

MML Write Access: status = GAIN(percent) built-in function, where percent is a real, $0 < x <= 1.0$.

HHP Read Access?: yes

HHP Write Access?: yes

$JERK -- Percentage of Maximum Jerk

Use:

This variable determines the rate of change of acceleration (jerk) as a percentage of the Maximum Jerk Allowable (system parameter 1180) that will be allowed for the next axis move.

Range:

`0.0 < $JERK <= 100.0`

Notes:

- If the range is exceeded, a run-time motion planning error will result.

Data Type: real

MML Read Access: direct

MML Write Access: direct

HHP Read Access?: yes

HHP Write Access?: yes

$LPLCO -- Array of 15 Long Integer Outputs

Use:

This array is used to transmit integer information to the host PLC via block transfers.

Range:

```
-79999999 < $LPLCO[n] < +79999999
```

Notes:

- Values are integer in the IMC 120. When transferred to the PLC, they are converted to 8 digit BCD. The most significant bit of the most significant word is the sign bit.
- Elements of $LPLCO are found in the following PLC blocks, words 3-8:

$LPLCO[n]	PLC Block No. (Words 3-8)
$LPLCO[1 to 3]	1
$LPLCO[4 to 6]	2
$LPLCO[7 to 9]	3
$LPLCO[10 to 12]	4
$LPLCO[13 to 15]	5

- You can use $LPLCO elements as general purpose integer variables in an MML application program if they are not used for communicating information to the PLC. This provides a practical way to conserve user RAM memory since $LPLCO elements use system RAM.

Data Type: integer

MML Read Access: direct

MML Write Access: direct

HHP Read Access?: no

HHP Write Access?: no

$OFFSET -- Axis Position Offset

Use:

This variable contains the position offset for the axis. Any moves programmed with a MOVE TO statement will have this value added to the programmed end-point. That is, the position to which the axis will move will be offset from the programmed position by this amount.

Range:

```
-real limit < $OFFSET < +real limit
```

Notes:

Data Type: real

MML Read Access: direct

MML Write Access: direct

HHP Read Access?: yes

HHP Write Access?: yes

$PHASE -- Initialize Master/Slave Phase for Ratioing

Use:

This variable is used to initialize the slave phase with respect to the master. In operation, the value of $PHASE is added to the Home Calibration Value (system parameter 1150) when a homing operation is performed.

Range:

`-real limit < $PHASE < +real limit`

Notes:

- This offset ($PHASE) will be applied anytime homing is performed, whether ratioing is to be done or not. Keeping this in mind, $PHASE may be used as a user programmable Home Calibration Distance in any application.
- The units of this variable are in electrical degrees of the feedback device. For example, if a resolver is used (4000 counts per electrical revolution), a $PHASE value of 90 will produce an offset of 1000 resolver counts (1/4 electrical turn).

Data Type: real

MML Read Access: direct

MML Write Access: direct

HHP Read Access?: yes

HHP Write Access?: yes

$PROBE__ARMED -- Probe Armed for Operation

Use:

This variable indicates whether the probe (either trigger probe or linear probe) is armed for operation or not.

Range:

```
$PROBE_ARMED = true -- probe is armed for operation
$PROBE_ARMED = false -- probe not armed for operation
```

Notes:

Data Type: boolean

MML Read Access: direct

MML Write Access: ARM__PROBE or DISARM__PROBE built-in procedures

HHP Read Access?: yes

HHP Write Access?: yes

$RATIO -- Master/Slave Ratio

Use:

This is a compound variable that specifies the value of the ratio that IMC 120 will follow a master axis.

Range:

This variable has 2 integer parts. The first part is the equivalent "slave gear teeth." The second part is the equivalent "master gear teeth."

```
-16383 < $RATIO < +16383 for each part
```

Notes:

- You can use a gearing analogy to think of the affects of $RATIO. If the value of $RATIO is set to (2,1) the slave axis will make 1 revolution for every 2 revolutions of the master axis (assuming that $RATIO__OVR is set to 1.0).
- The direction of ratioing may be reversed by setting either of the $RATIO components negative. For example, if the value of $RATIO is set to (-2,1) the slave axis will make 1 revolution in the opposite direction of the master every 2 revolutions of the master axis (assuming that $RATIO__OVR) is set to 1.0). If both of the components of $RATIO are set negative, the slave will follow the master in the same direction.
- Each component of $RATIO has a limit of ±16383. If you try to set either or both components of $RATIO outside this range, an error return on SET__RATIO (success = false) will be made and the values of $RATIO will be unchanged. Otherwise, a value of true will be returned.

Data Type: integer, integer

MML Read Access: GET__RATIO(s,m) built-in procedure, where s is the current value of the slave gear teeth component, and m is the current value of the master gear teeth component.

MML Write Access: success = SET__RATIO(s,m) built-in function, where s is the desired value of the slave gear teeth component, and m is the desired value of the master gear teeth component. True is returned if the values are acceptable (see Range, above). Otherwise, a value of false is returned.

HHP Read Access?: yes

HHP Write Access?: yes

$RATIO_OVR -- Ratio Override Value

Use:

This variable contains the ratio override value. Values in $RATIO are scaled by this value to obtain the target ratio. This variable is continually checked by IMC 120 software and any changes in its value that take place while a ratio move is in progress are immediately acted upon.

Range:

```
-9.9999 <= $RATIO_OVR <= +9.9999
```

Notes:

- A negative value is interpreted as reversing the direction of the slave axis.
- If a value greater than +9.9999 or less than -9.9999 is entered, the value will be changed to ±9.9999 respectively the first time that ratioing is enabled.
- If a value is entered that would make the slave axis output more than Output Voltage At + Max Speed (system parameter 2580) or less than Output Voltage At - Max Speed (system parameter 2590), the voltage output value will be limited to the corresponding limiting value.

Data Type: real

MML Read Access: direct

MML Write Access: direct

HHP Read Access?: yes

HHP Write Access?: yes

$RATIO__TYPE -- Type of Ratioing

Use:

This variable specifies the type of ratioing that will be performed when ratioing is enabled.

Range:

Possible values for this variable are:

- 0 = VEL__RATIO -- velocity ratioing, similar to open loop positioning. This type of ratioing is used to follow the master but never exceed the master speed. It could be used in applications where some slip is expected between the master and slave axes.
- 1 = POS__RATIO -- position ratioing, similar to closed loop positioning. This type of ratioing is used to closely follow the position of the master axis. The position of the master axis is followed beginning when ratioing is enabled.

Notes:

Data Type: integer

MML Read Access: direct

MML Write Access: direct

HHP Read Access?: yes

HHP Write Access?: yes

$RESULT -- Integer of RESULT Action

Use:

This variable represents the integer result from a global or local condition handler that has used the RESULT action.

Range:

`-integer limit < $RESULT < +integer limit`

Notes:

- $RESULT can only be written to by the condition handler RESULT action: RESULT int__exp

Data Type: integer

MML Read Access: direct

MML Write Access: RESULT action: RESULT int__exp, where int__exp is a valid integer expression

HHP Read Access?: yes

HHP Write Access?: yes

$SPEED -- Speed for Next Programmed Move

Use:

This variable is used to command the speed for performing the next programmed move.

Range:

`0 < $SPEED <= Maximum Programmable Speed`

Notes:

- The actual speed may be modified by $SPEED__OVR if $DISABLE__OVR is false.

Data Type: real

MML Read Access: direct

MML Write Access: direct

HHP Read Access?: yes

HHP Write Access?: yes

$SPEED__OVR -- Percentage of Speed Override

Use:

This variable contains the speed override percentage.

Range:

```
0 < $SPEED_OVR <= 127
```

Notes:

- When $DISABLE__OVR is false, the $SPEED variable is scaled by $SPEED__OVR to obtain the target move speed. This variable is continually checked by IMC 120 software and any changes in its value that take place while a move is in progress are immediately acted upon.

Data Type: integer

MML Read Access: direct

MML Write Access: direct

HHP Read Access?: yes

HHP Write Access?: yes

$SPLCO -- Array of 12 Short Integer Outputs

Use:

This array is used transmits information to the host PLC via block transfers.

Range:

```
-16183 < $SPLCO[n] < +16183
```

Notes:

- The values of elements in this array are smaller than the normal integers in MML, which range ±2147483647
- Elements of $SPLCO are found in the following PLC blocks, words 3-8:

$SPLCI[n]	PLC Block No. (Words 3-8)
$SPLCO[1 to 6]	6
$SPLCO[7 to 12]	7

- You can use $SPLCO elements as general purpose integer variables in an MML application program if they are not used for communicating information to the PLC. This provides a practical way to conserve user RAM memory since $SPLCO elements use system RAM.

Data Type: integer

MML Read Access: direct

MML Write Access: direct

HHP Read Access?: no

HHP Write Access?: no

$TERMTYPE -- End of Motion Type

Use:

This variable determines the termination type for the next programmed axis move. The value is displayed on the handheld pendant.

Range:

Possible values for $TERMTYPE are:

- 3 = FINE -- within fine in-position tolerance
- 2 = COARSE -- within coarse in-position tolerance
- 1 = NOSETTLE -- point at which following error begins to close-out
- 0 = NODECEL -- point at which deceleration must begin

Notes:

Data Type: integer

MML Read Access: direct

MML Write Access: direct

HHP Read Access?: yes

HHP Write Access?: yes

$UNITS -- Motion Units Type

Use:

This variable is used to select the units that programmed moves and positions will be interpreted as representing.

Range:
Possible values for $UNITS are:

- 0 = INCH/DEGREE -- program in inches (linear axis) or degrees (rotary axis)
- 1 = MM/REV -- program in millimeters (linear axis) or revolutions (rotary axis)

Notes:

- INCH and MM are only valid if the system is configured as a linear axis. DEGREE and REV are only valid if the system is configured as a rotary axis
- $UNITS may only be changed when the axis is at rest and there is no queued motion. The UNITS built-in function will return true if the conditions are met. Otherwise, the built-in returns false and $UNITS is not changed.

Data Type: integer

MML Read Access: direct

MML Write Access: success = UNITS(unit__val) built-in function, where unit__val is an integer expression equivalent to 0 or 1.

HHP Read Access?: yes

HHP Write Access?: yes

ALLEN-BRADLEY
A ROCKWELL INTERNATIONAL COMPANY

Intelligent Motion Control Division
747 Alpha Drive, Cleveland, Ohio 44143

Publication 1771-6.5.50 — May, 1988

PN 929647-01 PRINTED IN U.S.A.