

OPTIMO

PROGRAMMING MANUAL



U. S. Amada LTD
7025 Firestone Blvd.
Buena Park, Ca. 90621
(714) 739-2111

Table of Contents

CHAPTER 1		INTRODUCTION	
	PROGRAM COMPOSITION		1 - 1
	AVAILABLE SUPPORT		1 - 3
CHAPTER 2		MOTION INSTRUCTIONS	
	COORDINATE TYPES		2 - 1
	PATH INSTRUCTION		2 - 2
	Figure 1: Path types		2 - 2
	Figure 2: Path Cir		2 - 3
	Figure 3: PATH HOLE		2 - 4
	DYNAMIC INSTRUCTIONS		2 - 5
	Figure 4: Cut path using /FLY		2 - 5
	MOTION INSTRUCTIONS		2 - 6
CHAPTER 3		PROCESS CONTROL	
	LASER PROCESS INSTRUCTIONS		3 - 1
	OTHER		3 - 5
CHAPTER 4		HANDBOX OPERATION	
	HANDBOX DESCRIPTION		4 - 1
	Figure 1: Handbox		4 - 2
	Figure 2: Handbox key groupings		4 - 3
	Figure 3: Handbox mode switch		4 - 5
	Figure 4: Feedrate override		4 - 7
	HANDBOX OPERATIONS		4 - 10
	TCP DEFINITION (TOOL CENTER)		4 - 14
	Figure 5: TCP effect		4 - 15
CHAPTER 5		PART PROGRAMMING	
	OVERVIEW		5 - 1
	PROGRAM ORGANIZATION		5 - 2
	DATA INPUT		5 - 2
	USE OF SENSOR		5 - 3
	Figure 1: Approach speed		5 - 3
	PROGRAMMING AT THE HANDBOX		5 - 4

SPEED instruction	5 - 7
Motion control parameters	5 - 8
WORKDEF, WORK	5 - 10
OUTPUTS	5 - 13
LINKING PROGRAMS	5 - 14

CHAPTER 6	PROGRAM TESTING, EDITING
------------------	---------------------------------

PROGRAM TEST FUNCTIONS	6 - 1
BREAKPOINTS	6 - 1
SIMULATION	6 - 2
Instruction vs. Program Step Number	6 - 3
Figure 1: Effects of SIMUL on various instructions	6 - 4
PROGRAM TEST PROCEDURES	6 - 5
SINGLE Mode	6 - 5
CYCLE Mode	6 - 6
PROGRAM EDITING	6 - 7
Finding The Right Place	6 - 7
Selecting A Program Step	6 - 7
Deleting Instructions	6 - 7
Adding Instructions	6 - 8

CHAPTER 7	ADVANCED TOPICS
------------------	------------------------

LINKING PROGRAMS	7 - 1
GO	7 - 1
CALL	7 - 2
Figure 1: Program flow	7 - 2
Figure 2: Simple CALL instruction	7 - 3
Figure 3: Nested CALL instructions	7 - 3
SEQUENCE CONTROL WITHIN PROGRAMS	7 - 4
GOTO	7 - 4
Figure 4: Computed GOTO	7 - 4
GOSUB	7 - 5
RETURN	7 - 5
IF	7 - 5
PROGRAM OFFSET (SYSDEF INSTRUCTION)	7 - 6
Figure 5: Effect of SYSDEF	7 - 7
DECLARATIONS	7 - 8
MATHEMATICAL AND LOGICAL OPERATIONS	7 - 9
EXPRESSIONS	7 - 12
Figure 6: Math operators	7 - 12

Figure 7: Relational operators	7 - 12
ASSIGNMENTS	7 - 15
Usage of RML Variables	7 - 16
Assignment of REAL variables	7 - 16
Assignments of INTEGER variables	7 - 17
Assignments of LOGICAL variables	7 - 17
Assignments of Program I/O	7 - 17
READ ONLY variables	7 - 18
EXAMPLES OF USE OF VARIABLES	7 - 19

CHAPTER 8	OFF-LINE CONVERSIONS
-----------	----------------------

CUTTER COMPENSATION	8 - 1
INSTRUCTIONS	8 - 1
OFFSET AMOUNT	8 - 1
OFFSET DIRECTION	8 - 1
OFFSET HANDLING	8 - 2
OUTPUT FORMAT	8 - 2
BEGIN COMP	8 - 2
HALT COMP	8 - 2
CHANGE COMP	8 - 2
INCH/METRIC CONVERSION	8 - 3
Figure 1: File conversion	8 - 3
Figure 2: Format of examples	8 - 4
MANUAL OPERATION	8 - 5
DETERMINING LINEAR AND ROTARY UNITS	8 - 5
PREDEFINED VARIABLES	8 - 6
ROTARY TABLE	8 - 6
Figure 3: REAL variables	8 - 6
SYMBOL DECLARATIONS	8 - 7
CONVERSION PROCESS	8 - 8
ASSIGNMENTS	8 - 8
INSTRUCTION PARAMETERS	8 - 8
LINEAR AND ROTARY PARAMETERS	8 - 9
OTHER PARAMETERS	8 - 10
CONVERSION RATIOS	8 - 12
USING CONVERT FROM DOS	8 - 13
ERROR MESSAGES	8 - 14

CHAPTER 9	ADDITIONAL INFORMATION
-----------	------------------------

FUNCTION instruction.	9 - 1
-----------------------	-------

Figure 1: Files created using FUNCTION	9 - 2
Example of use	9 - 3
PASSING PARAMETERS AMONG MODULES.	9 - 5
Syntax of DEFINE	9 - 6
HANDBOX OPERATIONS	9 - 7
Figure 2: Handbox function keys	9 - 7
FUNCTION From the Handbox	9 - 8

CHAPTER 10	SAMPLE PROGRAMS
------------	-----------------

SAMPLE 1	10 - 1
SAMPLE 2	10 - 2
SAMPLE 3	10 - 5
INCH format	10 - 5
METRIC format	10 - 6
Ready For Compilation	10 - 7
Special Programs	10 - 8
Program 1	10 - 8
Program 2	10 - 9
Program 4	10 - 10

CHAPTER 11	APPENDICES
------------	------------

Appendix 1	11 - 1
INSTRUCTIONS	11 - 1
ATTRIBUTES	11 - 3
MATHEMATICAL FUNCTIONS	11 - 4
Appendix 2	11 - 5
SUGGESTIONS	11 - 6

CHAPTER 12	INDEX
------------	-------

Chapter 1

INTRODUCTION

PROGRAM COMPOSITION

A part-program consists of a series of instructions, which collectively describe a desired tool path, laser process, and any necessary machine instructions. Each instruction consists of the instruction name and one or more numerical or mnemonic parameters.

The Optimo uses a language called "Robot Machine Language", or "RML". Programs created or edited off-line must be in this format, which may be edited, printed, etc. as a text (man-readable) file.

Part- programs can be created on-line (by teaching), or off-line. Off-line programs must be put into the RML programming language. This may be done by a post processor from a CAD-CAM system or directly with a text editor. On-line programming may be done on the Optimo or at a teaching simulator.

Each program, before it is transferred to the control, must be "compiled" into "object" format. The "object" format includes all instructions from the RML "source" program in a more compact format. Comments from the source program are not included. A program which is in object format can be converted back to source format, for inspection or editing.

The conversion between source (RML) format and object (LMR) format is performed on an industrial computer, using software programs supplied with the system. For use of these programs, see the Software Manual.

The program is stored in the control (CRG) in battery-backed memory, called CMOS. Program length is limited to about 34,000 characters (current Optimos) in object format. The exact limit depends on the mix of instructions used in the program, and the version of "system software" in the CRG. If a program would otherwise be too long, the user may split the source program into two or more sections, and compile them separately. Each will then have its own program number in the control. It is often easier to test and edit shorter program segments than a single huge program..

When a program is run, the CRG first reads the object format program and creates an "executable" format version, which it then executes (runs). The conversion is performed only once, and the executable program is stored in CMOS with the original program's number+100. If the object version of a program is altered or deleted, the executable version is also deleted.

COMMENTS

Programs created or edited in RML format can have comments that make the program more readable.

A semi-colon (;) begins the comment line

A comment can also follow an instruction, provided that it is preceded by a semi-colon.

For example:

```
                ; THIS IS A COMMENT LINE
```

or:

```
SPEED 20        ; THIS COMMENT FOLLOWS AN INSTRUCTION
```

Notes

- 1: When compiling takes place, comments are not carried with the compiled program, though they remain in the source file
- 2: Comments and sequence numbers are not available from the handbox.

SEQUENCE NUMBERS

Some instructions refer to special identifiers, called "sequence numbers", for location within the part-program.

A sequence number may be any integer number between 0 and 9999. It cannot be a name.

For example, the instruction

```
    MOVE (1000, 1500, -250, -1.5, 0.25)
```

can be identified within the program by assigning it a "sequence number":

```
10  MOVE (1000, 1500, -250, -1.5, 0.25)
```

Sequence numbers may not be duplicated within a part-program.

AVAILABLE SUPPORT

This section covers the various types of assistance available, and what should be done before calling for help.

In case of trouble, please make sure you have read the applicable manuals / documentation first.

For assistance with cutting techniques, please be prepared to describe the material characteristics, any metallurgical requirements, and what type of contour or profile required.

For operational assistance, please have the related manuals at hand.

To report software difficulties, please note the series of operations leading to the problem. Note any changes made to the system prior to the difficulty. If a feature or function does not seem to work, make sure that the password / access level is correct.

Available technical support available from U.S. Amada's Laser Department includes the following:

- Programming information/assistance

- Operational assistance

- Laser cutting techniques

- Software support

- Maintenance procedures

Additional (fee-based) services available include

- On-site service

- On-site training

Another Amada-Group company, Amada Engineering and Service INC (AESI), handles spare parts for the Optimo. For new or refurbished repair parts, contact AESI.

Programming information/assistance

Operational assistance

Laser cutting techniques

Software support

Maintenance procedures

U.S. Amada, LTD.

7025 Firestone BLVD.

Buena Park, Ca. 90621

Voice phone: (714) 739-2111

FAX: (714) 670-1439

Repair parts

Amada Engineering and Service Inc

14921 East Northam Street

La Mirada, Ca. 90638-5798

Voice phone: (714) 670-2111

FAX: (714) 739-1948

MOTION INSTRUCTIONS

The MOVE instruction is used for all programmed motion.

All five coordinates (X, Y, Z, A, B) must be included. The system uses ABSOLUTE positioning only, not incremental. For systems having Inch/Metric select, the coordinates may be expressed in inches and degrees or in millimeters and radians. Examples in this manual will be labelled as to the units in use. The type of motion that occurs is determined by three types of parameters:

Static (COORD)

Kinematic (PATH)

Dynamic (SPEED, ACCEL, OVERSH).

The system has a program offset available as well (SYSDEF).

COORDINATE TYPES

Motion instructions may refer to either the tool tip position (COORD ABSOL), or to the end of the Z-axis column (COORD ROBOT) (intersection point of the A-axis rotary and B-axis rotary axes).

COORD ABSOL

Motion commands refer to the tool tip (nozzle tip) location in space, with rotations of A,B taken into account. All PATH types (PTP, LIN, CIR, HOLE) available in this mode. Used when contouring.

COORD ROBOT

Motion commands refer to the end of the Z-axis column. Only PATH PTP is available in this mode. COORD ROBOT is suitable for positioning only. Do not attempt to use for contouring.

The COORD instruction is modal within a part-program.

The "default" value of COORD can be set by parameter to be either COORD ROBOT or COORD ABSOL.

PATH INSTRUCTION

The path followed from one coordinate point to another is determined by the PATH command and COORD type. The following descriptions refer to the illustration on this page.

Available PATH types and the COORD types they are used with are listed below.

PATH PTP

(point to point motion) The machine moves from the initial point A to the target B according to an undefined path. The machine axes start and end their motion simultaneously, while the axis synchronization along the path is determined by the slowest axis. Use in COORD ROBOT for initial positioning, especially if large rotary moves are involved. In COORD ABSOL, the tool tip path is similar to LIN mode, but tip velocity is not controlled. Do not use when contouring.

PATH LIN

Linear path control, in COORD ABSOL only. The tool tip moves from the initial point C to the target D along a straight line in space. The tool tip speed is held constant. Rotary motion is synchronized with the traversing motion.

Example:

```
COORD ABSOL
```

```
PATH LIN
```

```
MOVE (1200.0, 1120.0, -150.0, 0.0, 0.0)
```

```
MOVE (2200.0, 1220.0, -250.0, 0.5, 0.5)
```

The tool tip will move in a straight line from initial position to the first target, (coordinate point X 1200, Y 1120, Z-150 mm, with the head at A 0.0, B 0.0 radians) then to the second target (coordinate point X 2200, Y 1220, Z-250 mm, with the head moving to A 0.5, B 0.5 radians)

Note 1: In COORD ROBOT, only PATH PTP is available. In COORD ABSOL, all PATH types are available.

Note 2: As with any CNC machine, a "safe approach" may be necessary so that the head clears fixtures and/or other obstructions.

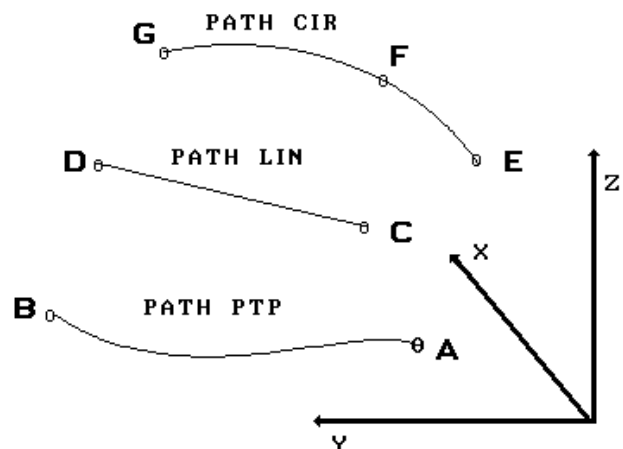


Figure 1: Path types

PATH CIR

Circular path control, available in COORD ABSOL only.

The tool tip follows an arc (circular path in space) defined by three points. The three points are:

- (1) initial position E
- (2) point on the arc F
- (3) end point G

The system moves from the initial point E to the target G, passing through point F. (see illustration on previous page) The tool tip velocity along the arc is held constant. Rotary motion is synchronized with the traversing motion.

The circular path needs one intermediate point in order to be carried out, specified by /VIA (..). If /FLY is used, it refers to the arc end point.

The normal restrictions apply to the use of /FLY. (see MOVE .. /FLY)

The format is:

MOVE (end point) / VIA (intermediate point)

- Example:

COORD ABSOL

PATH LIN

MOVE (1000.0, 1120.0, -150.0, 0.0, 0.0)/FLY

PATH CIR

MOVE (1200.0, 1120.0, -150.0, 0.5, 0.5) / VIA (1210.0, 1125.0, -125.0, 0.0, 0.0)

The tool tip (nozzle tip) will move from the initial position to the first target (coordinates X1000.0, Y1120, Z-150 mm, head at 0, 0 radians) following a straight line in space. It will then move to the second target (coordinate point X 1200, Y 1120, Z-150 mm, with the head at A 0.5, B 0.5 radians) by passing through the point defined by the coordinates after /VIA.

Notes:

1. The head angle in /VIA is ignored; only the coordinates of X,Y,Z are used. The head angle changes smoothly from initial position to end position. (each rotary axis takes the shortest path from starting position to ending position)
2. If the /VIA point is not specified, the path becomes a straight line type (same as LIN)

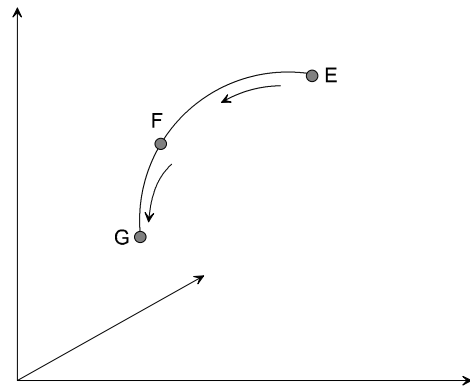


Figure 2: Path Cir

PATH HOLE

Circular hole normal to a defined centerline. Available in COORD ABSOL only. Defined by an initial position S, center point C, radius r, and a normal point N. The system computes a linear path from the initial point to the circle perimeter (call this location point A), then the circle is contoured counter-clockwise until point A is crossed. (viewed looking down at the part)

Format:

```
PATH HOLE (radius in mm)
MOVE (center point) /VIA (normal point)
```

Example:

```
PATH HOLE (25.1)
MOVE (1200.0, 1120.0, -350.0, 0.0, 0.0) / VIA (1200.0, 1120.0, -150.0, 0.0, 0.0)
PATH LIN
```

Notes:

1. The "normal point" is for computational purposes only. The system will not attempt to move to it or past it.
2. On the Teach Simulator (version 2.x software), Error 129 occurs when the program is run with the starting position on a line in space passing between the (center point) and (normal point). The Optimo does not have this limitation. Such programs may be created on the teach simulator, but not run on it.
3. The system does not open or close the laser shutter in the course of the HOLE type motion. Shutter open/close must be handled in the normal fashion with WORK ON before the hole, and WORK OFF after the hole.
4. The direction of traversed follows the right-hand rule, where positive rotation is defined as clockwise "looking into" the axis.

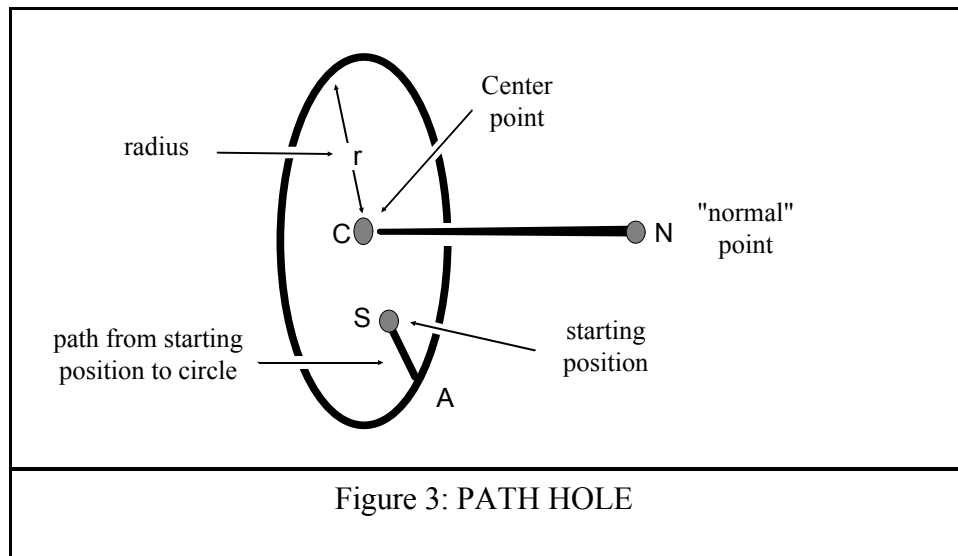


Figure 3: PATH HOLE

DYNAMIC INSTRUCTIONS

The motion dynamics are programmed with three instructions:

1. **SPEED [value]**. Specifies motion velocity as a percentage of the machine's maximum speed. The usual value for the Optimo MC series is 1000 inches/minute. The actual value is included with system documentation. **SPEED** is modal, from 0 to 100. The default value is 70.
Example: **SPEED 30** is 30% of 1000 inches/min, or 300 inches/min.
2. **ACCEL [value]**. Specifies the acceleration to be used during operation, as a percentage of the machine's maximum acceleration. **ACCEL** is modal, from 0 to 100. Its initialization value is 100. **NOTE**: a very low value of accel will result in little or no machine motion.
3. **OVERSH [value]**. Specifies the percent of programmed speed to maintain during /FLY passages. It is a modal parameter, and its value is between 0 and 100. The default value is 100.

Given the three targets A, B, and C one of which (B) is a "fly" passage: Call "Vb" the lower of the two running speeds in the lengths AB and BC. "OVERSHOOT" is defined as the percentage ratio between the speed in the bend (between A-B and B-C) and Vb.

If **OVERSH** is 0, the path deviation from B is zero, as B becomes a stopping point. The apparent dwell time is less, however, than a **MOVE** without "/FLY". If overshoot is equal to 100, the bend at B is traveled at the speed Vb (see illustration) The following discussion assumes **OVERSH 100**.

As speed at B is reduced, the path becomes closer to the programmed point. Speed reduction has the same result whether it is from the **SPEED** instruction, or the handbox % setting, or **OVERSH**, or **ACCEL**.

Speed, acceleration and "overshoot" are closely linked parameters. The term "overshoot" may not agree with conventional definitions used in automatic controls.

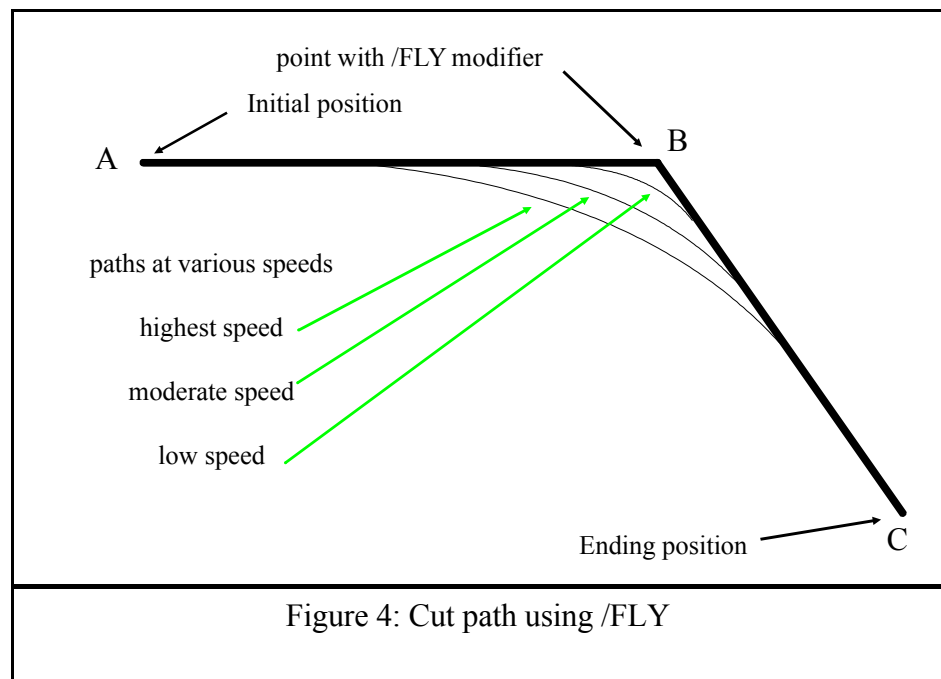


Figure 4: Cut path using /FLY

MOTION INSTRUCTIONS

The instruction MOVE is the real motion command. The axes move from current position to the target point, in accordance with the current static, kinematic and dynamic conditions, and according to the specified attributes of the instruction itself.

For example, by writing

```
MOVE (1200, 170.5, -118.3, 0.5, 0.12)
```

The machine moves to the absolute position (X1200, Y170.5, Z-118.3 millimeters) with the head angles (A0.5, B0.12 radians) in the current coordinate system.

Each coordinate is a "real" type number (with decimal point). The values can also be predefined real variables or constants.

MODIFIERS

Certain "attributes" or modifiers are used with the MOVE instruction. They modify the type of motion, or how (or when) it is begun or ended. These attributes can be written in any order without creating problems.

Commonly used attributes (modifiers) for the MOVE instruction are:

VIA, FLY, START

Also available are

TIME, END, NEXT

/VIA. Specifies the intermediate transition point of the circular path, or the point in a HOLE instruction which defines the normal axis. For examples, see the individual PATH instructions.

When programming from the handbox, the MOVE instruction is formatted according to current handbox PATH mode.

When programming off-line, or programming from the handbox with the handbox in the wrong PATH mode, the formatting may not be correct. If this attribute is used with types of path that do not require it, it is ignored. If it is omitted when needed, the motion will be executed according to a straight line path.

/VIA must contain all five coordinates, same as the first part of a MOVE instruction.

/FLY. Specifies that the machine will pass near but not necessarily through the target point while maintaining the percent of programmed speed set by **OVERSH** (default 100%). May not be used on the last move instruction before the end of a program, or on the last move instruction before a **COORD** instruction.

(see example on next page)

/TIME Sets a time limit on the **MOVE**. If the time expires before the move is complete, the control will perform a **GOSUB** to the sequence number specified.

Format: **MOVE** (target coordinates) **/TIME** (xx, yy)

Where xx is the time limit, and yy is a sequence number in the program.

/START Specifies that the motion can not start until some logical condition becomes true. Used to wait for an external event, such as an input coming on or turning off. Not normally used in part-programming.

Format: **MOVE** (target coordinates) **/START** (expression)

Where (expression) is any valid mathematical or logical expression. Usually just a logical input.

Example: **MOVE** (10.0, 10.0, 750.0, 0.0, 0.0) **/FLY /START** (IL[16])

/END Specifies a condition for motion stop. (other than arrival at target coordinates)

Format: **MOVE** (target coordinates) **/END** (expression)

Where (expression) follows the same rules as for **/START**.

NEXT Specifies that the next instruction in the program may be launched without waiting for completion of the **MOVE** instruction. The system will not necessarily reach the target coordinates, especially if the next instruction is also a **MOVE** instruction. Seldom used.

Format: **MOVE** (target coordinates) **/NEXT**

Example of MOVE () /FLY

```
SPEED          10
OVERSH         50
COORD          ABSOL
PATH           LIN
MOVE ( 1000.0, 400.0, -150.0, 0.0, 0.0)
MOVE ( 400.0, 400.0, -150.5, 0.0, 0.0) /FLY
MOVE ( 400.0, 1000.0, -150.0, 0.0, 0.0)
```

will use X400, Y400, X-150, A0, B0 as a target for the move but will pass only as near as it can achieve while maintaining 10%*50% of 30 m/min(1181.1 ipm). ($0.1*0.5*1181.1$ inch/min =59.1 inches/minute).

The nearness of passage is determined by the SPEED, ACCEL, and OVERSH, as well as the "sharpness" of the corner.

NOTES:

1. /FLY must not be used on the last MOVE on the part program, or prior to any operation which requires the machine to halt motion.
2. /FLY must not be used on a very short move (less than 1 mm) when the following move does not have the /FLY attribute.
3. /FLY should not be used on several short moves in sequence
Example: several moves in a row, each of less than .01 mm actual distance.

PROCESS CONTROL

Process control instructions perform tasks such as defining laser power ramping, opening and closing the shutter, performing timed dwells, and other functions.

LASER PROCESS INSTRUCTIONS

The laser power during cutting may be set by the WORKDEF instruction, and the shutter may be opened and closed with the WORK instruction. Several WORKDEFs may be active within a program, each with different laser power parameters.

WORKDEF

The WORKDEF instruction must be included in the part-program before any associated WORK instructions. The RML instruction format is:

WORKDEF n, p, (speed, power, delay, peak)

n = workdef type: 0 = no parameters used
 1 = parameters used
 2 = select laser program

BEAM ON/OFF

The laser shutter remains closed until the first request during a program for "beam ON". Assuming all conditions for "open shutter" are met, the shutter is opened and the beam is switched ON. When a "Beam OFF instruction is encountered in the program, the laser beam will be switched off but the shutter will remain open. The shutter is closed at program end and in HOLD condition when the operator door is opened. The shutter is also closed whenever any alarm condition occurs.

For beam on/off, WORK 0, 00 may be used. 1

WORK 0, 00 /ON opens the shutter using existing laser settings. WORK 0,00 /OFF closes the shutter.

The DELAY instruction may be used after "work on" to allow time for piercing.

If power modulation (either CW or pulse mode) is desired, WORKDEF 2,nn can be used to select a laser schedule prior to turning the beam on, or with the beam on.

WORK 1,nn /ON also performs BEAM ON. WORKDEF 1,nn may be used for combined pierce delay and power ramping control.

LASER SCHEDULE SELECT

This instruction permits selection of a laser schedule (a "laser schedule" is a setup file which has been created and stored on disk where the LCS can use it). The laser schedule determines pulsing/CW selection, power ramping (if used and what type, sets parameters), etc. When the WORK 2,nn/ON instruction is executed, the specified laser schedule is activated by the laser controller software. The shutter is not opened by WORK 2.

1 WORKDEF is optional for WORK 0,00.

POWER RAMPING

When the WORKDEF 1,nn instruction is used to control laser power, the laser power is altered according to the workdef parameters and computed tool tip speed.

(For systems with EFA-51 laser or EFA-emulation: To control power with workdef 1,nn, no EFA channel may be selected and modulation must be enabled. This means that OL[2] and OL[12]-OL[15] must all be OFF)

Ten sets of parameters may be selected (WORKDEF 1,00 through WORKDEF 1,09). The RML instruction format is:

WORKDEF 1, p, (speed, power, delay, peak, laser schedule)

p = WORK "process number" (0..9)

Speed, Power, Delay, and Peak can be numbers or "real" type variables or constants. Laser schedule is an interger from 1 to 63.

POWER (% of maximum power to output at the SPEED selected below)

SPEED (% of machine speed at which to output the POWER set above)

DELAY (time to dwell with shutter open at PEAK power, set below)

PEAK (peck-thru", or piercing power to use)

Program Number (refers to same "laser schedule" that WORKDEF 2,nn does)

Power ramping may also be performed using laser schedules. See the Operator Manual for more information on laser schedule creation and use.

WORK

The WORK instruction is used for laser process control. Depending on which type of workdef is being selected, it can open and close the shutter, start / stop the associated power ramping, or simply select a laser schedule. Note that every Workdef which has been activated with Work n,p /ON must be de-activated with a corresponding Work n,p /OFF prior to the end of program and prior to using the GO or CALL instructions.

Format::

WORK n, p /ON (OFF)

Where n, p correspond to a WORKDEF within the program, and ON or OFF defines whether the shutter is to be opened (ON) or closed (OFF).

Example 1:

```

WORKDEF 0,0
WORK 0,0 / ON
(Various motion instructions, etc)
...
WORK 0,0 / OFF

```

This example just opens the laser shutter and goes.
Suitable when material pierces and cuts easily at a constant power.

Example 2:

```

WORKDEF 1,0 (6.5, 80., 1.5, 95)
WORK 1,0 / ON
(Various motion instructions, etc)
...
WORK 1,0 / OFF

```

This example uses WORKDEF (power ramping) for power control and piercing.

Example 3:

```

WORKDEF 2,5 (15)
WORKDEF 2,6 (16)
WORKDEF 0,0
WORK 2,5 /ON
WORK 0,0 / ON
DELAY 0.5
WORK 2,6 /ON
(Various motion instructions, etc)
WORK 0,0 / OFF
WORK 2,5/ OFF
WORK 2,6 /OFF

```

This example uses laser schedules for power control and piercing

The delays are to allow complete piercing and to allow laser power to stabilize at the selected mode.

WARNING

Even though WORK 2,n /ON does not turn the beam ON, the corresponding WORK 2,n /OFF turns the laser beam OFF!

To know what parameters are really in use on any of the examples, the active or selected laser schedule or laser parameters must be examined.

Also affecting cutting performance is assist gas selection and type. When cutting with high-pressure gas it may be desirable to select low-pressure gas for piercing, then switch gases for cutting.

Remember that each WORK that gets turned ON must also get turned OFF prior to use of CALL or GO in the program, and prior to the program's end.

For information on laser control and operation, see the Optimo Operator Manual. For more information on power ramping, see the chapter on Part Programming in this manual.

OTHER

DELAY

Used to program a dwell time in a program..

Format: DELAY nnn.n

Where nnn.n is the time to dwell (in seconds).

ENDTASK

This instruction is normally used on other robots controlled by the CRG system. It is normally used in multi-tasking, to mark the end of one task. It may be seen at the end of some programs after they have been de-compiled, and it is seen on the handbox display when a program has run to it's end. This instruction is not needed in programs for the Optimo.

PAUSE

Halts program execution. When testing from handbox, continue execution by pressing the START key. When running from the console or gates, restart may not be possible, depending on version of system. Similar to STOP- differences only apply to other systems.

STOP

Halts program execution. When testing from handbox, continue execution by pressing the START key. When running from the console or gates, restart may not be possible, depending on version of system.

ERR

This instruction directs program execution (designated by a sequence number) in the event of a program error.

Format: ERR nn

Where nn is a sequence number in the program.

For examples, see the Amada special program listings in the Optimo Software Manual.

SETERR

Used to display an error status from a program. It does not create an error condition at the system level, merely sends a message to the handbox.

Format: SETERR nnnn

Where nnnn is a number from 1 to 9999. Displays a message on the handbox "ERROR nnnn"

NOTE: If this is to be used, contact Amada for error number assignments. (to avoid duplication and resulting confusion)

For examples, see the Amada special program listings in the Optimo Software Manual.

HANDBOX OPERATION

HANDBOX DESCRIPTION

GENERAL

The handbox provides display, editing, and other functions for the RoboPrima control (CRG). It is used for manual operations, calibration, programming, and other purposes. It has a 20 character vacuum fluorescent display, a mylar membrane keyboard with editing, control, and diagnostic functions, and an emergency-stop push-button.

The keys are organized into groups according to function. On the following two pages the handbox is illustrated, and the keyboard grouping is shown and numbered. The following text refers to these group numbers.

HANDBOX DISPLAY (1)

The display, in the CRG's default mode, shows :

P = XX	S = XXXX	I = XXXXXXXX
--------	----------	--------------

where:

P = Program number

S = Step number

I = Name of instruction in program Pxx , step Snn

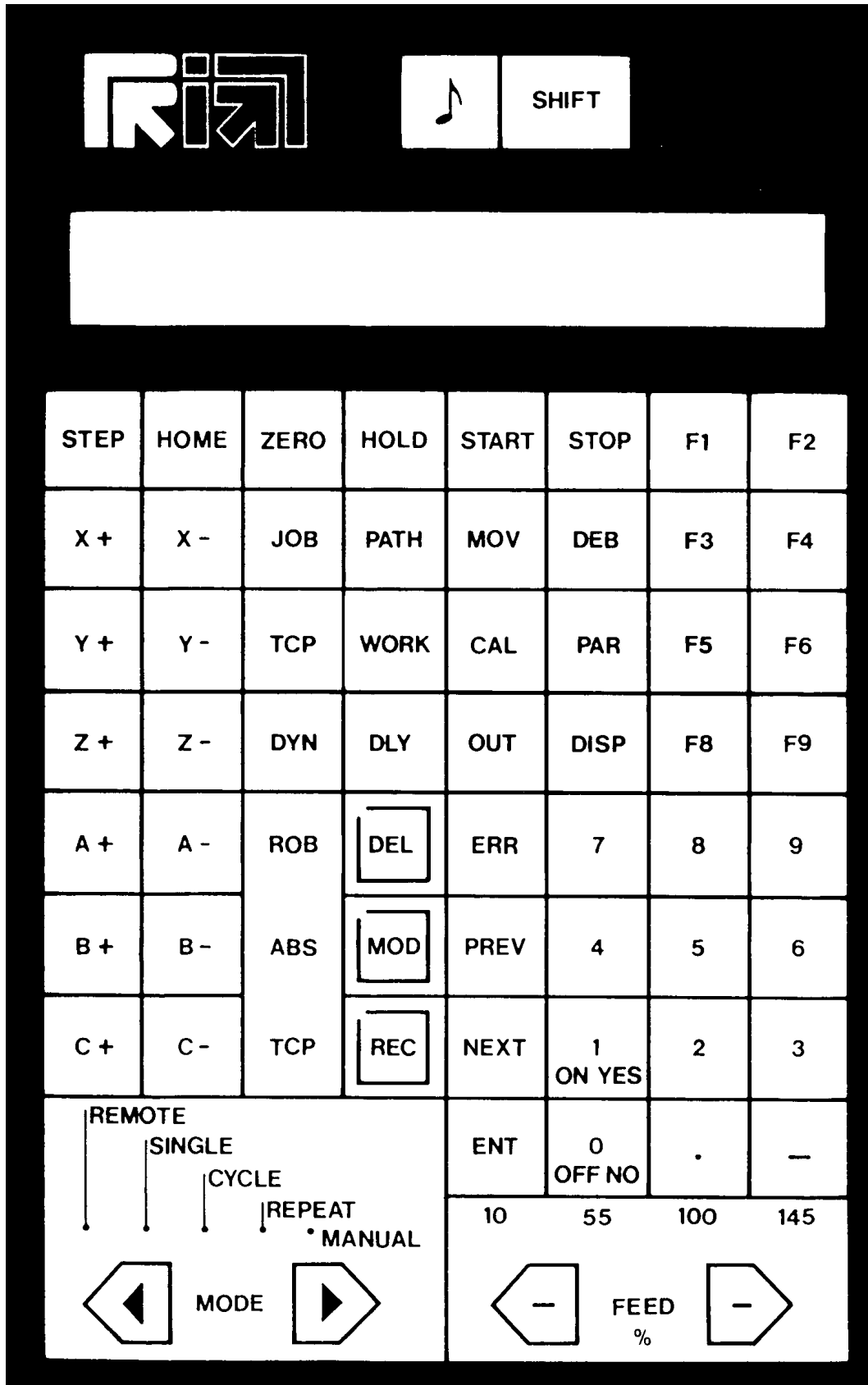
NOTE: If an error condition exists, the error code will be displayed instead of the above. Depress the ERR key to clear the error message from the display (this does not clear the error status).

SHIFT (2)

The SHIFT key is used with DEL, PREV, and NEXT keys for selecting additional operations. An LED indicates if the function is enabled.

MUSIC NOTE (3)

The "music note" key activates a "beep" which sounds each time a key is pressed.



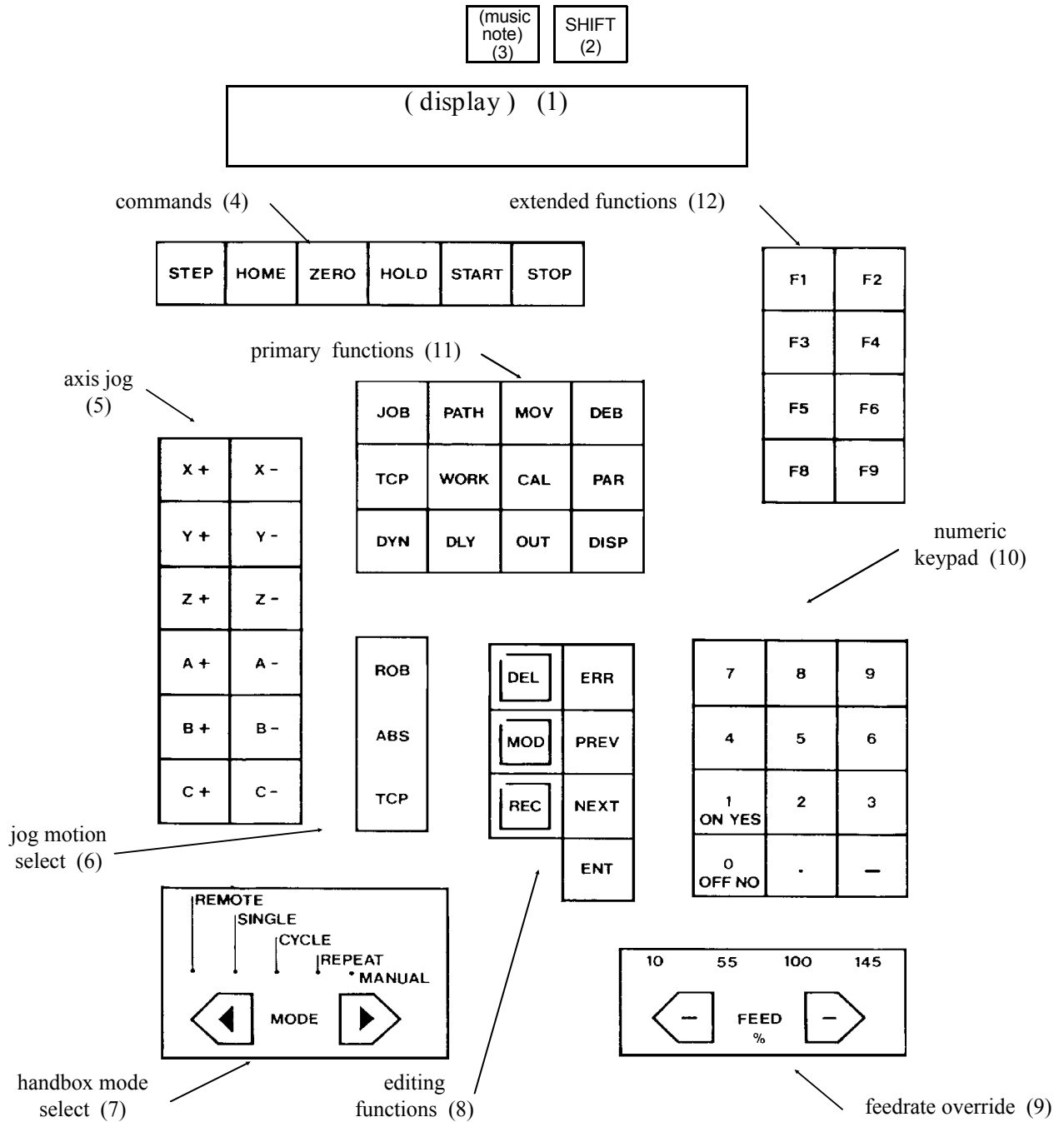


Figure 2: Handbox key groupings

COMMANDS (4)

These keys are used in creating and testing part-programs. None of them are enabled when the handbox is in REMOTE for normal machine operations.

KEY	ENABLED MODES	FUNCTION
STEP	MANUAL	used in program testing
HOME	MANUAL	jog axes to predefined position
ZERO	MANUAL	perform ZERO AXIS procedure
HOLD	SINGLE / CYCLE / REPEAT	halt program execution after current instruction finishes
START (1)	MANUAL	access GO, CALL instructions
START (2)	SINGLE / CYCLE / REPEAT	begin program execution
STOP	SINGLE / CYCLE / REPEAT	stop program execution

AXIS JOG (5)

This keypad is only enabled in manual mode. It consists of 12 keys for the forward and backward movements of five machine axes. The motion stops when the key is released. The "C" axis is not capable of manual jog operations.

JOG MOTION SELECT (6)

These keys select the axes system reference in manual jog operations. Each key has an LED to indicate active mode. ROB motion is the default.

MODE	ACTION
ROB	each axis moves independently
ABS	tool tip holds constant position when rotating A, B
TCP	X, Y, Z axes "carried" with head as rotations are made

HANDBOX MODE SELECT (7)

The MODE switch allows selection of handbox mode. The active selection is indicated by an LED.

Selection is performed by pressing either of the arrows, so that the LED below the desired mode is illuminated.

MODE	ENABLED OPERATIONS
MANUAL	Any manual operations, such as jogging the machine, teaching part-programs, turning outputs ON or OFF, etc
REPEAT	Program test. At program end, restarts execution at beginning of that program.
CYCLE	Program test. At program end, execution ceases.
SINGLE	Program test. The START key must be pressed for each program step
REMOTE	Normal system operation from console or gates.

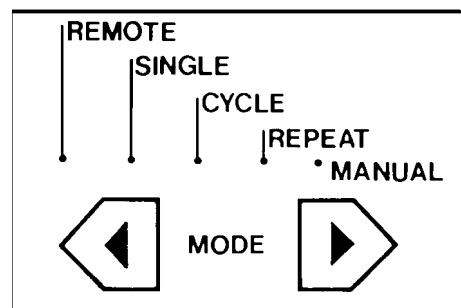


Figure 3: Handbox mode switch

EDITING FUNCTIONS (8)

Used in part-program creation and modification.

KEY	ENABLED MODES	FUNCTION
DEL	MANUAL	clear data field or delete selected instruction
shift DEL	MANUAL	delete selected program
MOD	none	not used
REC	MANUAL	record instruction into active part-program
ERR	ALL	acknowledge and clear error message from display
PREV	MANUAL	select previous instruction in current program
shift PREV	all except REMOTE	select previous part-program
NEXT	MANUAL	select next instruction in current part-program
shift NEXT	all except REMOTE	select next (higher numbered) part-program
ENT	MANUAL	activate instruction (and associated mode) in the control and prepare it for recording into a part-program.

FEEDRATE OVERRIDE (9)

Selects a multiple of commanded feed speed. Active during all modes of operation except the zero axes procedure. The speed can be changed "on the fly" during machine motion and program execution.

Ten steps are available. For axis jog the speeds selectable are: 0.05, 0.3, 1, 3, 9, 15, 21, 27, 33, and 40. % of the machine's maximum speed. (0.6, 3.5, 12, 35, 106, 177, 248, 318, 389, 472 inches per minute on 30 meter/minute machines)

In all other modes (program run) the selectable speeds are: 10, 25, 40, 55, 70, 85, 100, 115, 130, and 145 % of programmed speed.

Note: the CRG defaults to a speed of 70% at the beginning of each program.

NUMERICAL KEYPAD (10)

Keys from 0 to 9, decimal point, and negative sign (positive sign is assumed and need not be expressed).

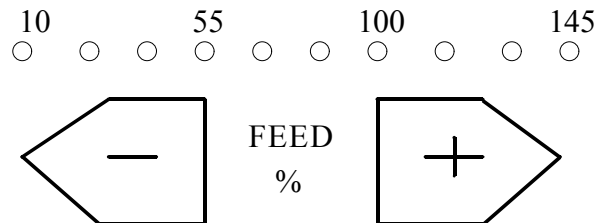


Figure 4: Feedrate override

PRIMARY FUNCTIONS (11)

Used for part-program selection and various programming, setup, and calibration operations. LEDs indicate the enabled function. (see table)

KEY	ENABLED MODES	FUNCTION / INSTRUCTION
JOB	All except REMOTE	Selection of part-program for creation, editing, or running
TCP	MANUAL	TCP instruction for nozzle length compensation
DYN	MANUAL	COORD instruction and handbox mode selection (ABSOL / ROBOT)
PATH	MANUAL	PATH instruction, display current PATH mode (PTP, LIN, CIR, HOL) and select new mode
WORK	MANUAL	WORK WORKDEF instructions Laser beam ON/OFF and parameter setup
DLY	MANUAL	DELAY instruction
MOV	MANUAL	SPEED instruction, /FLY
CAL	MANUAL	access to machine parameters/ calibrations
OUT	MANUAL	LET instruction (program outputs for sensor ON / OFF, etc)
DEB	MANUAL	BREAK function (stop program at a selected step)
PAR	none	not used
DISP	none	not used on Optimo

EXTENDED FUNCTIONS (12)

F1 - F8 and DEB - PAR - DISP keys used for various programming, program test, and setup operations. LEDs indicate the enabled function. (see table)

KEY	ENABLED MODES	FUNCTION / INSTRUCTION
F1	MANUAL	WEAV -related (not normally used)
F2	MANUAL	OVERSH instruction
F3	MANUAL	not used
F4	MANUAL	not used
F5	MANUAL	not used
F6	MANUAL	SIMULATE function (dry run to a selected program number & step)
F7	MANUAL	WEAV - related (not normally used)
F8	MANUAL	GOSUB (Optimo only)

HANDBOX OPERATIONS

The Optimo can be operated from the handbox in manual mode, which allows jogging one or more axes, or a part-program may be executed in the single-step, cycle, or repeat modes.

ABSOLUTE ZERO AXES

When the system has not been zeroed, the only motion available is JOG mode on the handbox.

The system must be Zeroed before entering into programming mode. The system may be zeroed (or re-zeroed, if desired) by the following procedure.

1. Select MANUAL mode (when in STOP or HOLD conditions)
2. Press the ZERO key. The following mask is displayed :

ZERO AXIS?

Depressing the YES key causes the system to automatically move the axes towards the machine reference points. While the zero operation is in progress, the ZERO AXES lamp remains illuminated and the "ZERO AXES" flashing message is displayed on the handbox indicating that the operation is in progress.

The Z-axis zeroes first, then the A,B axes, then the X,Y axes.

After each group of axes has zeroed, they are moved to the "mechanic offset" position. The next group of axes is then zeroed. When all axes have been zeroed, the handbox prompt returns to the default

Note: When the ZERO AXIS procedure is activated from the console the system performs the zero axis procedure, which leaves the machine at the "mechanical zero" reference coordinates. then launches program #4 (home return). Program #4 then moves the machine to the standard home position.

MANUAL MOTION

The machine is moved ("jogged") by pressing one or more of the axis jog keys. Each axis has two buttons, one for forward axis motion (+), and one for backward axis motion (-).

In jog mode linear and angular feed speed is controlled by the FEED % keys. More than one axis may be "jogged" at the same time if desired. Motion will continue while the button(s) are held down.

In manual mode, there are three types of motion available:

1. ROBOT AXES (ROB).
2. ABSOLUTE AXES (ABS).
3. TOOL AXES (TCP).

ROBOT AXES

In ROB mode, each "jog" motion moves the selected axis only without affecting the position of any other. When A-axis or B-axis motion is commanded, the nozzle tip will describe an arc in space and the other axes will remain motionless.

ABSOLUTE AXES

In ABS mode, each "jog" motion refers to tool tip motion, as opposed to raw axis motion. The result for moving X, Y, or Z is the same as those in ROBOT mode. In the case of A-axis or B-axis motion, the X,Y,Z axes will "compensate" to maintain the nozzle tip at the same ("X", "Y", "Z") coordinate point in space as the rotary axes are moved.

TOOL AXES

In TCP mode, the cutting head "carries around" the reference system. When the cutting head is in "home" position the machine reference system and tool reference system are coincident. As long as the A and B-axes are not rotated from the home positions, the two reference systems are effectively the same. As the head is rotated, the TCP reference system is rotated with it. When all axes are at the home position, the directions of X+, Y+, and Z+ can be marked on the head (with pencil, or on pieces of tape). These marked directions indicate TCP motion directions at any head orientation.

HOME KEY

The HOME key, when pressed, moves all axes toward the home position. The motion is ROBOT type.

PART-PROGRAM SELECTION

Each OBJECT part-program stored in the CRG memory is identified by a program number (1 to 99).

Part-program selection is done through the JOB key while in any mode except REMOTE. When in STOP or HOLD condition the following is displayed:

P = XX	S = XXXX	T = X	M = X
--------	----------	-------	-------

the fields present in this mask are:

P = program number (1 to 99)

S = step number (1 to 999)

T = (not used on Optimo)

M = (not used on Optimo)

The second digit of P field flashes to indicate readiness for input. NEXT key moves input to other fields. ENT key completes the operation, and ERR aborts it. The DEL key clears a field of previous data.

The T and M instructions are not implemented on the Optimo system.

Part program selection can be done in any mode except REMOTE. After selecting the desired program through the JOB mask, depress ENT key and the following mask will be displayed:

P = XX	S = 0000	I = #####
--------	----------	-----------

P = selected program number

S = program instruction step

I = current instruction

(see following page)

Notes on program selection:

1. When no STEP number is selected, step "zero" is displayed, with asterisks in the INSTRUCTION field.
2. In the "I" field, only the instruction "name" is displayed. (MOVE, SPEED, etc)
3. In manual mode, the system will permit the "selection" of a program that does not exist. To confirm existence, two approaches may be used.
 - a) Use SHIFT PREV, SHIFT NEXT. The system will only step between existing programs, so if the program number in question is not currently used, the system will step past it to one which is in use.
 - b) After pressing ENT to select the program, press REC. If the program number is already in use, the last program step will be displayed. Otherwise, the program will be created, and the display will show "Pnn S0001 I= ENDTASK"

The system is ready to run the program, in the selected mode (CYCLE, SINGLE or REPEAT), when the START key is pressed. At end of each step, "S" will increment and "I" will show the current instruction.

Part-programs may also be selected by using the SHIFT/PREV and SHIFT/NEXT keys. SHIFT,NEXT displays the next program in memory (next higher number). SHIFT,PREV selects the previous program in memory. If no program is selected, then SHIFT/NEXT selects the first number in memory, while SHIFT/PREV selects the last.

PART-PROGRAM DELETION

To Delete a part-program:

SHIFT/DEL is used to delete part-programs. Select the part-program to delete (using JOB or SHIFT NEXT or PREV), then depress SHIFT, DEL keys. The display shows DELETE Y/N

Press the "yes" button to delete the program, or "no" to cancel the delete program request.

TCP DEFINITION (TOOL CENTER)

The Optimo is calibrated so that, in ABSOL mode, all coordinates refer to the actual position of the "tool tip" (the tool tip is located 0.8mm from the nozzle tip, coincident with the beam and with the sensor turned OFF). This assumes that the nozzle is set at the same focal point as the one used when the system was originally calibrated. When the focus height is changed from nominal, the TCP instruction is used to make tool compensation adjustments in the system.

The TCP instruction needs six values, of which four will always be zero on Optimo. The first three values are distances, in tool coordinates, of the current nozzle tip position from the normal (originally calibrated) position. The second three values are distances, in tool coordinates, from the normal tip position to a reference point. The relationship between these two values sets the tool "axis" to be used in TCP mode.

From the handbox, the tool dimensions (in millimeters) are inserted using the TCP key:

TCP coordinates

X0 = 0.0	(always zero on Optimo)
Y0 = 0.0	(always zero on Optimo)
Z0 = 0.0	(distance of the actual nozzle tip from the normal location).
X1 = 0.0	(always zero on Optimo)
Y1 = 0.0	(always zero on Optimo)
Z1 = -5.0	(coordinates of the point that defines the TCP "plus" direction)

Before inserting a new value, the old value must be cleared using the DEL key.

The NEXT and PREV keys are used to move from one mask to the next.

When the desired values are entered, the procedure is completed by pressing the ENT key, making the TCP value active in the control. The TCP value will affect any program that uses COORD ABSOL which is run with TCP active, unless it has a TCP instruction in it. (the most recent TCP instruction executed becomes active)

If desired, the machine may be jogged in ABS or TCP to see the new TCP value's effect. When ready, use the REC key to record the TCP instruction into the active part-program.

Example of TCP use in RML:

TCP (0.0, 0.0, 3.0 / 0.0, 0.0, -6.0)

Alternate format

TCP (0.0, 0.0, 3.0, 0.0, 0.0, -6.0)

This places the new TCP (with respect to the calibrated nozzle tip location) coincident with the beam and shifted along Z axis 3 mm. (toward the focusing lens) Its orientation is defined by a line from the first point to the second.

Note: When jogging Optimo in TCP axes, the "+/-" orientation is affected by the TCP coordinates. For correct jog motion in TCP, make sure that Z0 is always numerically larger than Z1. ($Z0 > Z1$)

See the OPTIMO OPERATOR MANUAL for more information.

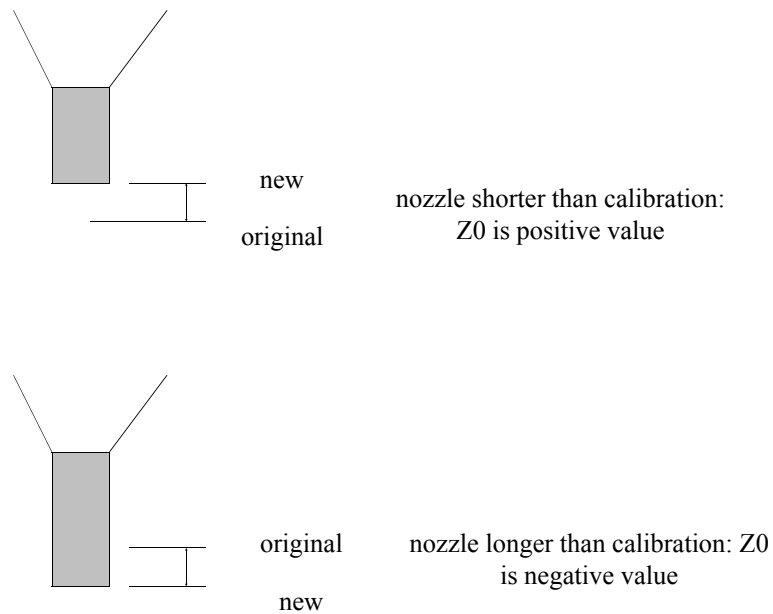


Figure 5: TCP effect

PART PROGRAMMING

This section concentrates on Teach mode programming. This is the process of programming from the handbox, using the system's facilities to record instructions into a part-program. The program structure and content of "taught" programs can provide a guide to off-line programming.

OVERVIEW

The operator uses the JOG keys to move the machine axes to each position needed in the program, and records those points into the program, along with other instructions as necessary.

The general sequence of operations for teach programming are:

1. Prepare and position the fixture/part to be programmed.
2. Put the system into PROGRAM mode at the console.
3. Take the handbox into the work area, and proceed to create the program.

Note: it is good practice to make at least three reference marks on the part fixture, and teach those points as a separate program. This program should be transferred to the PC for safekeeping, in case the part needs to be run in a new location.

4. As portions of the program are completed, they may be test-run to catch errors early.
5. If the program is long and/or complex, use the MMI's Scheduler to create a "project" and transfer the program to the PC for safekeeping. As sections of the program are completed, use the scheduler to get updates into the PC. This helps protect against accidental loss or damage.
6. When the program is complete, and runs properly, use the MMI's Scheduler to create a "project" (if you haven't already) and transfer the program to the PC for safekeeping. The project should include the laser schedules used and other setup information at this time.

NOTE: SYSDEF must be cancelled before performing any teach- related operations.
Cancel SYSDEF by executing a SYSDEF instruction without any parameters, as follows:

SYSDEF

Program 2 in the CRG (Z-UP) performs this function. A separate program can also be created with only the SYSDEF instruction, if the entire Z-up operation is not desired.

After a program cancelling SYSDEF is run, teaching operation may proceed.

PROGRAM ORGANIZATION

The typical part-program will have the following:

1. setup instructions (outputs for laser select and sensor, WORKDEF, etc.)
2. SPEED, COORD, PATH, MOVE instructions for positioning
3. WORK ON to begin cutting
4. SPEED, COORD, PATH, MOVE instructions for contouring as required
5. WORK OFF
6. SPEED, PATH, etc for retract away from the part.

Items 2-6 are repeated as necessary for each individual feature to be cut on the part. (SPEED, PATH for retract and approach are often the same)

DATA INPUT

When an instruction key is pressed, the display will show the first "mask" pertaining to that key. A mask generally includes the instruction name and one or more data fields. The keys PREV and NEXT are used to move from one field to the next, or (in some cases) to select instructions. When the correct data has been selected or input, the ENT key is used to complete that input. When data entry is complete for an instruction, and if a program is currently selected, the instruction name will flash on the display. This indicates that the instruction is ready to be recorded into the part-program, by pressing the REC key. If desired, the instruction may be aborted (before activation) by pressing the ERR key. If the instruction is already flashing (ready for the REC key), the ERR key has no effect.

Recording Instructions

When an instruction has been activated by pressing the ENT key, the instruction name will flash on the handbox display. If that instruction is to be recorded in the active program, then press the REC key. If the flashing instruction is not needed in the program, then the next instruction may be selected by pressing the appropriate instruction or jog key.

Aborting An Instruction

To abort an instruction such as MOVE (get rid of the flashing instruction prompt), press the MOV key, so that SPEED is displayed. Then press the ERR key, and the display return to default.

Numeric Data Fields

If a numeric field needs to be changed, then use PREV or NEXT to select it. (the selected, or "active" field flashes) If the number displayed has a decimal point, it must be cleared by pressing the DEL key. The new value may then be entered using the numeric keypad. If the numeric field does not have a decimal point, then DEL is not needed. As new numbers are entered, they appear at the right of the field, and the other numbers are shifted off of the left side. The DEL key may be used, if desired.

NOTE: During some numeric inputs, the ERR key will merely create a "garbage character" in the numeric field. In that case, use DEL to clear the field, then key in the correct value, if needed.

- To clear a flashing instruction display, press the MOV key or DLY key, then press ERR.

USE OF SENSOR

The C-axis sensor is really necessary anytime focus and/or nozzle tip height from the part is to be closely maintained. Before starting to program, verify that the correct tip is installed, that the tip standoff is set, etc. See the Operator Manual and/or the Maintenance Manual for information on these topics.

The sensor is activated by turning OL[11] ON. This is done with the OUT key. See the OUTPUT section in this chapter for more information.

RECOMMENDED SPEEDS

For rapid traverse well away from the workpiece, speeds up to 70% may be used. Closer to the workpiece, 30% - 50% should be used. On the approach move that actually brings the sensor tip to the work surface, do not let the tip approach the workpiece faster than the sensor can respond. A reasonable speed is 10% on MC systems. Higher speeds may be used only if the approach path is diagonal, so that normal velocity is 10% or less. (see illustration below)

Actual feedrates along the workpiece must be determined by program testing and process testing. Usually speeds from 5% to 15% will be used in cutting typical metal parts.

When moving away from the workpiece, speeds of 20% or less are recommended, until the head is well clear of the part.

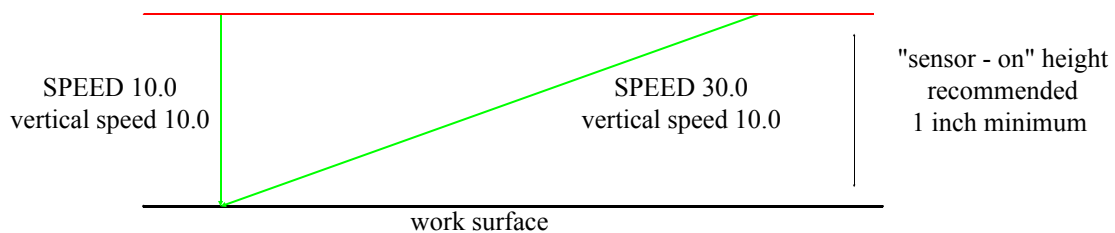


Figure 1: Approach speed

PROGRAMMING AT THE HANDBOX

HANDBOX MODES

When an instruction is activated, it becomes effective for further operations at the handbox. It affects related program instructions RECORDED WHILE THE INSTRUCTION MODE IS ACTIVE. It will not alter the way a previously recorded instruction (or program) executes. It is crucial when recording a program that the "handbox modes" match the instructions actually in the program.

An instruction is activated on the handbox when the ENT key is pressed. It is not necessary to record the instruction when only setting handbox mode. In the following section, it will be noted how (or if) an instruction affects the handbox mode.

COORD instruction

The COORD instruction determines whether each MOVE command refers to tool tip coordinates or to the Z-axis column coordinates. COORD ROBOT is the system default, and is suitable for positioning only. COORD ABSOL refers to the tool tip (0.8 mm from nozzle tip) and is used for all contouring. When the COORD instruction is ENTERed, it selects which values the system uses when RECOding MOVE commands.

The COORD instruction is accessed with the DYN key. The current handbox mode is then shown, either COORD ABSOL or COORD ROBOT. Use the PREV or NEXT key to select the desired mode, and press ENT to activate it. To record the instruction in the active part-program, press the REC key.

PATH instructions

The PATH instruction (along with COORD) determines the type of motion followed. PATH PTP is used for positioning only. PATH LIN is used for positioning and straight-line contouring, and PATH CIR is used for circular arc contouring. PATH HOLE is used for cutting round holes on flat or nearly flat surfaces. All contouring must be done in COORD ABSOL.

The PATH instruction is accessed with the PATH key. The current handbox mode is then shown. The PREV and NEXT keys are used to select the desired path type, and ENT activates it. To store the instruction, press REC.

System Defaults

The system defaults for COORD, PATH, and SPEED are determined by system calibrations. See the Service Manual for information on calibrations.

The P&W machine is set up for COORD ABSOL, PATH LIN, SPEED 70 as of 1/93.

MOVE INSTRUCTIONS COMPARED			
PATH TYPE	COORD used with	Function	Coordinate Format
PTP	ROBOT or ABSOL	positioning	MOVE (end point)
LIN	ABSOL only	positioning or contouring	MOVE (end point)
CIR	ABSOL only	contouring only	MOVE (end point) / VIA (point on arc)
HOLE	ABSOL only	contouring only	MOVE (center point) /VIA ("normal point")

NOTE: In PATH PTP and PATH LIN only one point is required for each MOVE instruction, while in PATH CIR and PATH HOLE two points must be recorded for each MOVE instruction. Under normal conditions, the handbox handles this automatically. When editing a program, or resuming after a break or shutdown, make sure that the handbox is in the correct mode. Otherwise the MOVE instructions may not be in correct format for the PATH in that part of the program.

Path PTP is for positioning only. In PATH PTP, each time a set of coordinates is recorded, it makes a single MOVE instruction. In COORD ABSOL, the trajectory looks linear, but without tool tip speed control.

Path LIN may be used for positioning or contouring. It is available in COORD ABSOL only. PATH LIN requires one set of coordinates per MOVE instruction, like path PTP. The handbox may be in PTP or LIN mode to record points for path LIN. This instruction is selected and recorded just like PATH PTP.

In PATH PTP or PATH LIN the step counter should increment each time a point is recorded. If it does not, the handbox is in the wrong PATH mode.

Path CIR is normally used for contouring only. Like LIN, it is available in COORD ABSOL only. In CIR, two points must be recorded for each MOVE instruction. The first point must be on the arc (not at the start or end point), and the second IS the end point. Head angle is ignored on the first point (/VIA point). During program execution the head motion is synchronized from starting position to ending position.

When recording points for CIR type motion, the handbox must be in PATH CIR mode. (if the handbox is in PTP or LIN mode, the points will not be formatted correctly, and the system will move to each point with LIN type motion).

Once the handbox is in PATH CIR, the indicated program step ("step counter") will increment every second time a MOVE is RECORDED. The operator can use this to assure that the coordinates are being recorded properly. (When recording a midpoint, the step counter should not increment. When recording the end point of an arc, the step counter should increment.)

PATH HOLE is used to cut round holes of any size, on a flat surface at any angle. The instruction requires two points for each MOVE instruction; the first to indicate hole centerpoint, and the second must be perpendicular (normal) to the surface from the centerpoint. The instruction is selected (as are all path instructions) through the PATH key. PATH HOLE is selected using PREV or NEXT keys, and ENT is pressed. The display now shows

RAD = _____

The numeric value displayed will be the last programmed radius, or zero. To change the radius, press DEL, then key in the new value. When the indicated radius is correct, press ENT. The handbox display will then display the instruction name (PATH) flashing. To add the instruction to memory, press REC. The handbox will now display (flashing)

ENTER CENTER NORMAL

Now the two positions (center point and normal point) must be recorded. The flashing message continues until the first of the points is recorded. Then, as the machine is moved toward the second point, the usual flashing MOVE is shown. The step counter does not increment until the second point is recorded. WORK must be turned ON before the MOVE, and OFF afterward. Approach and retract must be done in PTP or LIN, so for each HOLE, the following sequence of commands is necessary:

```
MOVE (approach to starting point)
WORK 1,00/ON           ;select working process as appropriate
PATH HOLE (25.0)
MOVE (center point)/VIA (normal point)
WORK 1,00/OFF
PATH PTP
MOVE (away to set up for next cut)
```

NOTES:

1. When programming at the Teach Station: If the starting position of a hole is at the hole center or on the line connecting the center point and the normal point, an error will occur when the program is run at the Teach Station. The program will run correctly on Optimos with system software v.4.2 and later.
2. If the hole seems to be elliptical ("out-of-round"), then the "normal point" was not correct or the surface is skewed. Another symptom of this is excessive C-axis motion while moving around the hole.
3. /FLY should not normally be used on the positioning move or the hole move.
4. The handbox must be in HOLE mode when recording points for path hole. The radius, however, only appears at the PATH instruction, not the MOVE instruction. When altering points for a hole, it is not necessary that the radius shown be correct, only that the handbox PATH mode is correct.

5. If the hole appears out-of-round, but not elliptical, the SPEED may be too high for the size of hole. Try reducing the cutting speed. (for large changes, don't forget to adjust power too)
6. If the programmed radius is zero, CRG error 348 results.

SPEED instruction

The MOV key is used to select the SPEED instruction. This instruction affects motion in both COORD ABSOL and ROBOT, in all path types. The default SPEED is 70%. To insert a SPEED command, press the MOV key. The display (default) is

SPEED 70.0

To correct the numeric value; press DEL, then key in the correct value. Press ENT to activate the instruction, and REC to add it into the current program.

The MOV key is also used to access various modifiers for the MOVE instruction.

MOVE instruction

The MOVE instruction commands the machine to move from current position (at the time the program is executed) to the coordinates ("target point") specified. The type of motion executed depends on the active COORD and PATH. Each time the robot is jogged, with a program active, the display will flash "MOVE". This means that its ready to record that instruction, with current coordinates, into the active program.

The RML format for MOVE is:

MOVE (x-coordinate, y-coordinate, z-coordinate, a-position, b-position)

Each coordinate is a "real" type number, variable, or constant.. When programming from the handbox, each MOVE instruction gets all five coordinates in the form of real numbers, corresponding to machine position when the REC key is pressed.

Motion control parameters

Several modifiers are available for the MOVE instruction. These affect machine motion and/or program execution, so they must be used properly. The most commonly used is /FLY, but /START and others are available when needed. (for further information, see chapter 2)

The /FLY modifier instructs the system to maintain speed past one target point in the program to the next point. The actual speed past the point will be affected by several factors, to be covered later. The /FLY modifier is added by the system to each MOVE instruction recorded when the "fly mode" is active.

The "/FLY mode" is only a handbox mode; it is not modal in a program. The programmer/operator must be aware of /FLY status when programming, to avoid undesired results. Any point where the machine does not need to stop is a candidate for use of /FLY. Any point where the machine MUST stop must not be recorded with /FLY.

USE OF /FLY IN A PROGRAM	
OPERATION	Use of /FLY
positioning in space	optional (normally yes)
DELAY	NO
positioning to piercing point (before a WORK 1,nn /ON instruction)	NO (as this instruction includes a DELAY)
positioning to piercing point (before a WORK 0,00/ON instruction)	OK if no delay is used (can't predict exactly where shutter will open)
sharp corner	NO (unless making an "outside loop" or setting OVERSH =0 for that corner)
transition along cut path	YES
transition between COORD types	NO (FATAL error)
last MOVE in a program or last MOVE before a GO or CALL	NO

To activate "fly mode", press the MOV key. When SPEED is displayed, press NEXT. The display will show FLY = #, where # is a number zero or one. Zero means "fly mode inactive", while "FLY = 1" means that fly mode is active. To activate, press 1, then ENT. The display will flash SPEED, but the instruction need not be recorded- as soon as ENT is pressed, "fly mode" is set to the value entered. As soon as a jog key is pressed, the flashing MOVE display will return. Make sure that the last MOVE instruction in a part program is recorded with fly inactive. For further information, see chapter 2.

Other modifiers are accessed through the MOV key. When SPEED is displayed, use NEXT to step past FLY to the other available masks. As with FLY, a "1" means "ON" and "0" means "OFF". Pressing ENT prepares the system to add the selected modifier to the next MOVE instruction recorded.

The /START, /TIME, /END, /NEXT modifiers, unlike /fly, are not modal. The first MOVE instruction recorded after selecting a modifier will have the modifier added to it. Subsequent instructions will not have it. These modifiers are seldom used in part-programming.

REMINDER: the handbox/CRG must be in the correct modes (COORD, PATH, /FLY, etc) so that the move will be recorded correctly. If the handbox is in COORD ROBOT, then the coordinates recorded will be those of the Z-axis column (not the tool tip).

OVERSHOOT parameter

This parameter is accessed with the F2 key. The mask is:

OVERSH __

Press DEL to clear the old value, and key in the new value. Press ENT to make it active. and REC to record it into the active part-program. OVERSH does not affect any other handbox modes or operations. For information on the effect of this instruction, see chapter 2.

WORKDEF, WORK

On systems with the ARROW laser, the shutter stays OPEN during automatic operation, and the laser beam is turned on and off. Systems with the EFA-51 laser must open and close the shutter to start and stop cutting. The part-program commands are the same for either system. Systems with the EFA-51 may not have access to laser schedules. Systems with ARROW lasers do not have "channels" to select.

The laser power during cutting may be set by WORKDEF, and the shutter may be opened and closed with the WORK instruction. Several WORKDEFs may be active within a program, each with different laser power parameters.

NOTE: WORKDEF is required before using the WORK instruction in a part-program.

The WORK key (in the PRIMARY FUNCTIONS group) is used to access both instructions. By pressing WORK the following mask is displayed:

TOOL = n	PROGR = nn
----------	------------

Where:

TOOL = laser process parameters (0 = disable, 1 = enable, 2 = schedule select)

PROGR = WORK process number (0..9)

To just turn the beam on, use TOOL = 0 and PROGR = 0

PREV and NEXT are used to move between fields, ENT enters the information and displays the following mask:

WORK = XXX

Where:

XXX can be selected (using the PREV and NEXT keys) from:

DEF (WORKDEF instruction)

ON (open shutter, execute defined dwell, etc)

OFF (close the shutter)

WORKDEF 1

When DEF is selected, the following masks are available (if TOOL = 1)

POWER (% of maximum power to output at the SPEED selected below)
SPEED (% of machine speed for POWER above)
DELAY (time to dwell with shutter open at PEAK power, set below)
PEAK (call this the peck-thru", or piercing power to use)
PROG (laser schedule number to use, 1-63)

Use PREV and NEXT to display the masks, DEL and the numerics to enter the data. Press ENT when the data is complete. The following will be displayed, flashing.

WORKDEF

Press REC to store the instruction in the active part-program.

WORKDEF 2

When DEF is selected, the following mask is shown (if TOOL = 2)

PROG (laser schedule number to use, 1-63)

Use DEL to clear the field and the numerics to enter the laser schedule to activate when the matching WORK instruction is turned ON. Press ENT to activate the process. The following will be displayed, flashing.

WORKDEF

Press REC to store the instruction in the active part-program.

WORKON

When ON is selected and ENT is pressed, the following is displayed:

START = 0

Normally, press ENT again, and the following is displayed, flashing:

WORKON

Press REC to record the instruction in the active part-program.

WORKOFF

When OFF is selected, and ENT is pressed, the workoff instruction is displayed, flashing:

WORKOFF

Press REC to record the instruction in the active part-program.

CONTROLLING POWER WITH WORKDEF

To control power with workdef, the laser must be in CNC with the pulser active. Ten sets of parameters may be selected (WORKDEF 1,00 through WORKDEF 1,09). The RML instruction format is:

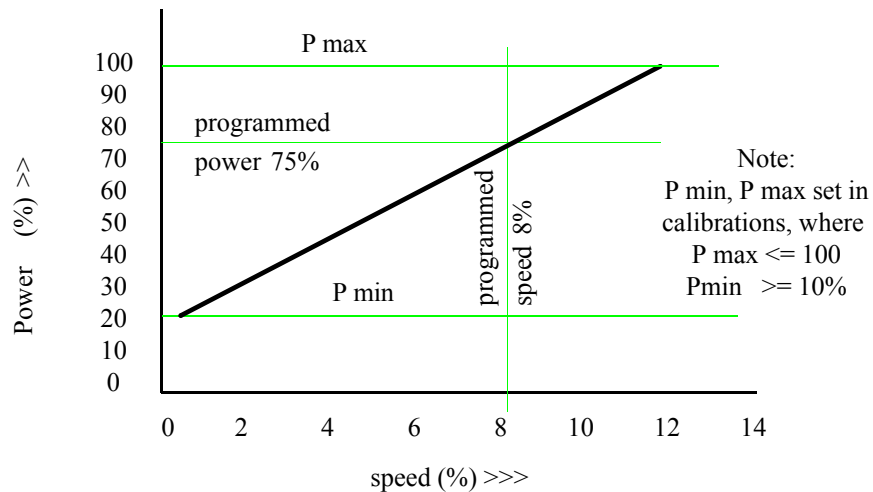
WORKDEF n, p (speed, power, delay, peak)

n = parameter enable: 0 = no parameters

1 = parameters used

2 = laser schedule select

If "parameter enable" = 1, then the laser power parameters may be input, where speed, power, delay, and peak are numbers or "real" type variables or constants.



Power ramping using WORKDEF 1,00 (8.0, 75.0, 1.0, 50.0)
(Dwell 1 second at 50% power, then contour using 75% power at 8% speed)

OUTPUTS

The laser channel select, sensor ON / OFF, and other functions are controlled by program outputs. These outputs are actually sent to the PLC, which then checks to see if the action is permitted. If the action is permitted, then the PLC turns it's corresponding output(s) ON (or OFF, as appropriate). The program outputs are accessed using the OUT key (in the primary functions group) Press the OUT key to display the following:

OUT nn = x

Where nn is a value from 01 to 99, and x is either 0 (zero) or 1 (one). PREV and NEXT move between fields, and the numbers may be keyed in without deleting the old values. When the values are correct, press ENT. The following is displayed:

P = nn S = nnn I = LET

To record the instruction into the active part-program, press REC.

Output #	action
8	select cutting gas 2
9	select cutting gas 3 (option)
11	Turns sensor ON

Notes:

1: Only one assist gas may be selected at a time.

LINKING PROGRAMS

The GO and CALL instructions are available in manual via the START key.

GO Jumps to another program and continue operation.

CALL Jumps to another program and returns when that other program is finished.

To use either instruction, select MANUAL on the handbox, then press START. The following appears:

ISTR GO

if GO is to be used, press ENT. Otherwise, press NEXT.

ISTR CALL

When ENT is pressed, the following appears:

PROG nn

use the numeric keys to enter the program number to CALL or GO to, and press ENT.

One of the following will appear (whichever one that was selected)

P = __ S = ____ GO

or

P = __ S = ____ CALL

Press REC to record the instruction into program memory.

PROGRAM TESTING, EDITING

Program testing is the process of determining the correctness of tool path, motion type and smoothness, and process parameters. Tool path correctness may be visually checked while teaching, by "playing back" portions of the taught program. Motion type and smoothness can be checked the same way. The final and (and most important) test for fixture and program correctness is in measurement of the cut part. It must always be remembered that errors in fixturing or forming the part will affect final accuracy.

PROGRAM TEST FUNCTIONS

The BREAK and SIM functions are available for program testing, debug, and correction on Optimo. They are usually used in program mode, to check some portion of the part-program without having to run the entire program each time it is tested.

BREAKPOINTS

The BREAK function will interrupt part-program execution at a designated program step. The program may then be modified and/or continued, as desired. To use BREAK, select MANUAL on the handbox and press DEB. The following will be displayed:

BREAK = OFF

Use PREV or NEXT to select between ON and OFF.

When ON is selected and ENT is pressed, the following is displayed:

STEP = 000 PRG = nn

Where nn is the current program.

The PREV and NEXT keys step between fields, and values may be keyed in directly. When the values are correct, press ENT. The BREAK function is then active.

A part-program may now be selected and run. The system will run normally until the program number and step are reached that were specified in BREAK. The control will stop after executing that instruction, just as if the HOLD key had been pressed.

Notes:

1. The breakpoint is not cancelled at the end of the program. It remains active until set OFF with the DEB key, or the control is powered down.
2. Only one breakpoint can be specified at a time, and replaces the previous breakpoint.

SIMULATION

The F6 / SIM function allows a program, or a portion of it, to execute in "simulation mode". The control executes the part program, starting at the beginning of the program and continuing up to but not including the step specified in the SIMUL instruction. During the simulation, no motion occurs and the working processes stay inactive (shutter closed, etc.). It is usually run from the handbox and is essentially "dry-run" mode.

The portion of the program (if any) beginning with the instruction number specified will be executed in real time, NOT in SIMUL mode.

To use the SIM function, select MANUAL and press F6. The following is displayed:

SIM	PRG nn	STEP 0000
-----	--------	-----------

Where nn is the program number to simulate. (need not be the currently selected program) STEP is the first step to execute in normal mode.

PREV and NEXT step between the fields. The values may be keyed in directly, or DEL may be used to clear a field. When the values are correct, press ENT. The function will then be active, and when the program is run, it will run in simulation mode until the selected step number is reached. Normal execution will then commence.

When program execution is started, the system will run in simulation mode until the selected program number and step number is found. The selected step will be run in REAL mode (not simulation mode). WARNING: While the legal values of <step> are from 1 to 9999, program control MUST pass to the selected program, or the system will stay in simulation mode. To clear simulation mode, select a program for simulation which can be run, and then run it. In extreme cases, the control may need to be powered down.

SIM and BREAK may be used together. This permits stopping at a particular step while still in simulation mode, to begin real execution in SINGLE mode instead of CYCLE..

PROGRAM (if specified) must be an integer value from 0 to 99. The current program is the default if no program number is specified.

During program simulation, modal parameters and input/output images are evaluated.

NOTES:

- 1: Make sure that a clear path is available between the machine's initial position and the first MOVE instruction to be executed after the simulation is complete.
- 2: Make sure that the first MOVE command executed is in PATH PTP or PATH LIN. Other path types will give undesired results.
- 3: The term "step" does not refer to sequence numbers, it refers to the actual line of program code being executed. Each program line increments the step counter by one except for the DECLARE statement.
- 4: Optional Rotary Table System (FUNCTION 99):
The optional Rotary Table is not moved during simulation, and is not updated after exiting simulation mode. The display of commanded position for the RTS may not be correct.

INSTRUCTION	STEP NUMBER
DECLARE XAXIS = VR[0]	0
DECLARE YAXIS = VR[1]	0
DECLARE ZAXIS = VR[2]	0
DECLARE RAPID = VR[64]	0
DECLARE APPROACH = VR[69]	0
DECLARE FEED = VR[74]	0
XAXIS = 500.0	1
YAXIS = 375.0	2
ZAXIS = -50.5	3
RAPID = 70.0	4
APPROACH = 20.0	5
FEED = 12.0	6
COORD ABSOL	7
PATH PTP	8
SPEED RAPID	9
MOVE (XAXIS, YAXIS, ZAXIS, 0., 0.) / FLY	10
SPEED APPROACH	11
.	

Instruction vs. Program Step Number

SIMUL is normally used to test motion-type programs. For programs which depend on variable external conditions, such as input tests, analog input readings, etc. SIMUL is of less help. An example is the "home return" program (#4), which looks at current axis position to determine a return path sequence.

The following table lists the effects of simulation on various instructions:

INSTRUCTION	OPERATION
MOVE	<ul style="list-style-type: none"> -START attribute is ignored -END attribute is ignored -TIME-OUT is not activated -The movement is not physically performed
WORK	<ul style="list-style-type: none"> -START attribute is ignored -END attribute is ignored -TIME-OUT is not activated -The shutter is not opened.
ZERO	<ul style="list-style-type: none"> -ZERO AXES is considered performed, but the operation is not physically done
DELAY	<ul style="list-style-type: none"> - Not performed
Logical, analog physical outputs	<ul style="list-style-type: none"> - Not performed (images used) - Physically updated when normal operation resumes
TIMER	<ul style="list-style-type: none"> - Not performed
Rotary Table System (option)	<ul style="list-style-type: none"> not moved position is not updated when system exits simulation mode.
Laser schedule select	<ul style="list-style-type: none"> - Not performed - Physically updated when normal operation resumes

Figure 1: Effects of SIMUL on various instructions

PROGRAM TEST PROCEDURES

This section will only cover program testing from the handbox, in Program mode. For other operations, see the Optimo Operator Manual. Three modes are available at the handbox for program testing; SINGLE, CYCLE, and REPEAT. When a program is under test, the operator may switch from one mode to another, whenever the control is stopped at the end of an instruction. The START key is used to start or resume execution of a program. The STOP is used to halt execution of a machine instruction immediately. (usually to halt machine motion) The HOLD key will allow the current instruction to finish execution, then inhibit further program execution. The START key resumes program execution after use of the STOP or HOLD key.

SINGLE Mode

SINGLE mode is used to test-run a program one step at a time. To test a program from its beginning:

Select the desired program (JOB key)

Select SINGLE mode

Set the FEED% to desired override. (if ANY uncertainty exists, use the lowest setting)

Press the START key. The controller (CRG) will display and execute the first program step.

In SINGLE mode, the START key must be pressed for each program instruction. Any previous instruction must finish execution before the START key becomes effective.

- a. Execution time of non-MOVE instructions is not affected by the FEED% setting. Any instruction which does not include motion or a program dwell will execute very rapidly.
- b. During execution of a MOVE instruction, motion can be halted by pressing the STOP key. Motion may be resumed by pressing START again.

In SINGLE mode no effect is seen of MOVE ()/FLY. This is because the system stops at each target point, and the /FLY is always computed using actual velocity at program run time (which becomes zero in SINGLE mode at the end point)

It is recommended that MODE be changed (from SINGLE to CYCLE or the reverse) only at the end of a program instruction.

CYCLE Mode

To test a program from the beginning in CYCLE mode: (continuous execution)

1. Select the desired program (JOB key)
2. Select CYCLE mode.
3. Set the FEED% to desired override. (if ANY uncertainty exists, use the lowest setting)
4. Press the START key. The controller (CRG) will begin program execution.

In CYCLE mode, the START key must only be pressed once to run the entire program.

a. Execution time of non-MOVE instructions is not affected by the FEED% setting. Any instruction which does not include motion or a program dwell will execute very rapidly, and may not even be displayed on the handbox display.

b. During execution of a MOVE instruction, motion can be halted by pressing the STOP key. Motion may be resumed by pressing START again.

In CYCLE mode the effect of MOVE ()/FLY is seen. However, the /FLY passage is always computed using actual velocity at program run time, so the actual path is affected by the program speed and the FEED% selected at the handbox, as well as the OVERSH and ACCEL values active.

During program execution, use the HOLD key to stop execution at the end of a motion block or program instruction.

NOTE:

The first time a program is run after creation or revision, a delay will occur upon pressing START. (in SINGLE, the delay only occurs the first time).
This is caused by the control reading the program and creating an "executable" version of it.
The handbox will display "COMPILING" during this operation.

PROGRAM EDITING

Program editing from the handbox consists of testing the program, deleting undesired or incorrect program instructions, and inserting new instructions as needed. This process can be demanding, as the operator must make certain that the system is in the correct "handbox modes" when inserting new steps. When a program is created from the beginning, only the /fly status needs attention. When editing a program, the COORD, PATH, and /FLY modes must all be set properly.

Finding The Right Place

It can be difficult to identify the correct location within a part-program for modification. Generally, you can STOP or HOLD in the area of interest, reduce feedrate %override, and single-step to the offending location in the program.

The above procedure may still not tell the full story. To get a listing of the program with step numbers, use the MMI's Scheduler to transfer it to the MMI as a project (or part of one). When the transfer is done, the most recent "listfile" will be your program (in metric format) with line numbers, step numbers, and all program instructions.

Selecting A Program Step

A program step is "selected" when it is displayed on the handbox display. Generally, the program may be run in CYCLE and/or SINGLE up to the desired point, or the JOB and/or the PREV / NEXT keys may be used.

If a MOVE instruction is to be changed or added, it is best to run the program up to the point of interest. The SIMUL and BREAK functions may be used if desired to aid the process. If a MOVE instruction is only to be deleted, then machine position is unimportant. As long as an instruction can be correctly identified, it can be deleted.

The JOB key may be used to select the program itself and the step number to display. If the desired step is not near the very beginning or end of a program, and you know the step number, use the JOB key. If the desired location is near the end of a program, or you wish to find the end of a program, use the JOB key and select the program number, then press ENT, then REC. That displays the first vacant step at the end of the program and "ENDPRG"

When a program is selected, the PREV key is used to move forward in the program toward the end. The PREV key is used to move toward the beginning of the program. These keys move one step in the program each time they are pressed.

Deleting Instructions

To delete an instruction, simply select the instruction and press the DEL key. The handbox display shows:

DELETE Y / N ?

Press the 1/YES key to delete the instruction, or the 0/NO key to not delete it.

Adding Instructions

When an instruction is recorded, it goes into the program BEFORE THE DISPLAYED INSTRUCTION. The previously displayed instruction is "pushed ahead" of the new instruction, and is displayed again. Each Instruction after the one added has its step number incremented.

To add or change a non-move instruction, the machine position is unimportant. The proper program location must be selected, and program steps may be added and/or deleted at will.

If a MOVE instruction must be added or changed, the machine must be moved to the new location. The usual technique is to run the program up to the area of interest, then HOLD or STOP with the instruction showing that is to be changed. Select MANUAL and perform any needed deletions. Make sure that the handbox is in the correct COORD, PATH, and /FLY modes.

PATH PTP, PATH LIN

Then jog the machine to the desired position, and press REC. In PATH PTP or PATH LIN the step counter will increment and the new instruction is added to the existing program.

PATH CIR

In PATH CIR two points must be recorded to make a single MOVE instruction. The first point must be on the arc (between the previous instruction's end point and the added instruction's end point) and the second becomes the end point. When the first point is recorded the step counter will not increment. When the second point is recorded, the step counter will increment. Make sure to reselect the correct PATH before recording any other type of move instructions.

CAUTION: When testing/editing a MOVE in PATH CIR, either let the old instruction execute and then delete/re-teach it, or re-select and re-run the program after teaching the new move. Otherwise the old move (still in EXECUTABLE) will not start from its normal location, and will follow an unpredictable trajectory. (the old EXECUTABLE will create a circle connecting the new taught end point to the old end point and passing through the old /VIA point)

PATH HOLE

In PATH HOLE, the procedure is very similar to that of PATH CIR- two recorded points make a single MOVE instruction. Make sure the handbox is in COORD ABSOL and /FLY is OFF, then select PATH HOLE. When the flashing ENTER CENTER NORMAL prompt appears the points may be recorded. When the first point is recorded the step counter will not increment. When the second point is recorded, the step counter will increment. Make sure to reselect the correct PATH before recording any other type of move instructions.

ADVANCED TOPICS

In the RML language, various features are available which add power beyond the simple COORD, PATH, SPEED, MOVE commands. "Variables" of several types can have values assigned, to be used in the part-program as needed. Names can be assigned these variables, for better readability and easier use. "Constants" can also be created, where a value is used repeatedly in a program and doesn't change within it. Mathematical and logical operations are available, and can use constants and/or variables. Execution sequence within the program can be controlled as well, using GOTO and GOSUB.

One example of the use of symbols and mathematics is the "home return" program, program #4. It checks machine position before moving, then raises the head (Z-up), then brings the axes to a safe approach point, then home. A simpler example is the "shifting" of a program, by adding a local offset (part fixture offset) to a table offset and using the result in the part-program's SYSDEF.

LINKING PROGRAMS

GO

The GO instruction is used to chain (link) part-programs together. It is available from the handbox, using the START key in MANUAL mode. The RML format of the instruction is:

GO nn / C

Launches program #nn. The program number is an integer value. It may be a number, a constant or a variable. The "/C" means to run the linked program in CYCLE mode

The GO instruction does not modify the data area: ALL variables are preserved from one program to another.

NOTE: Each variable should be initialized before use. Unless a variable is being used to transfer data from one program to the next, it should be initialized in each program.

CALL

The CALL instruction is used to execute a "program gosub". The CALL instruction may be used when you need to structure your program using pre-built modules, or are in need of a program which is greater than 32 Kbytes in its executable format. In the latter case the program can be broken in small sub-programs callable from a main one. Each sub-program is a program with length up to 32 Kbytes in its executable format.

The CALL instruction is available from the handbox, using the START key in MANUAL mode. The RML format of the instruction is:

CALL nn

Causes program #nn to be run. When that program terminates, execution is returned to the calling program. The program number must be an integer value. It may be a number, a constant or a variable.

The CALL instruction does not modify the data area: ALL variables are preserved from one program to another.

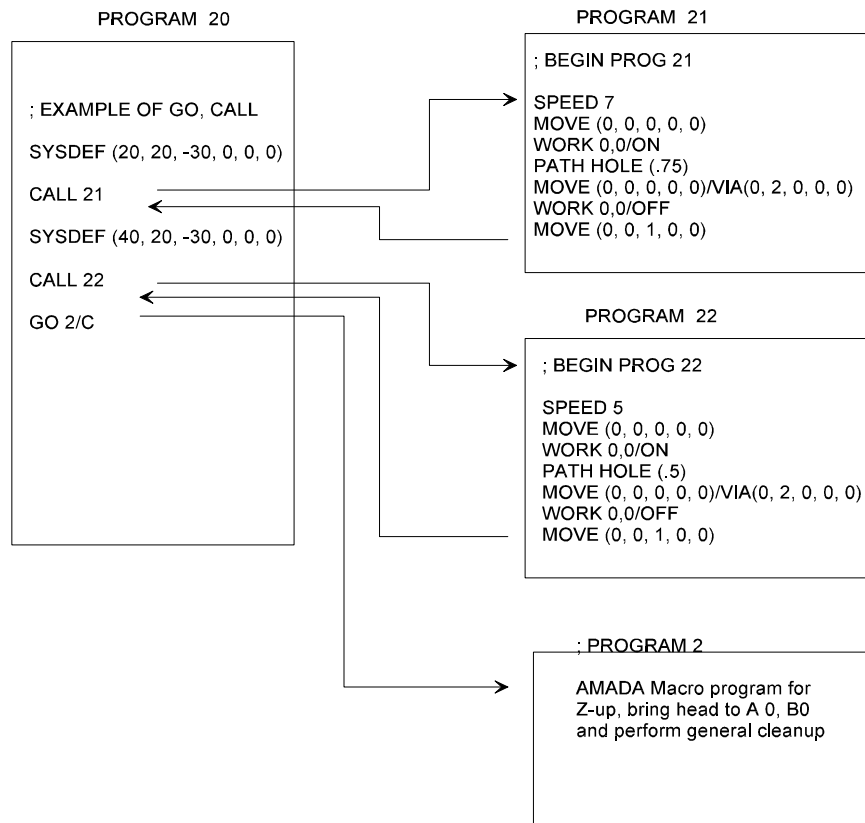


Figure 1: Program flow

The CALLED program may also call another program. They may be nested up to four levels deep.

The fourth level program called must already be in EXECUTABLE format.

NOTE: Each variable should be initialized before use. Unless a variable is being used to transfer data from one program to the next, it should be initialized in each program.

When a CALL pg_n (program number) instruction is executed, the CPU will load the code of program pg_n from the CMOS (storage memory) to the working memory, thus overwriting the actual running program. It then executes the newly loaded program. The called program will be first compiled if the executable format is not available. When the instruction ENDTASK is reached the CPU will load the calling program (MAIN) on its working memory and will resume execution of the instruction after the CALL.

Because of this program overlay procedure the CALL has a relatively long execution time. If faster execution time is needed and size is not a problem (32 Kbytes Max.) it is advisable to use the FUNCTION instruction (see section on FUNCTION).

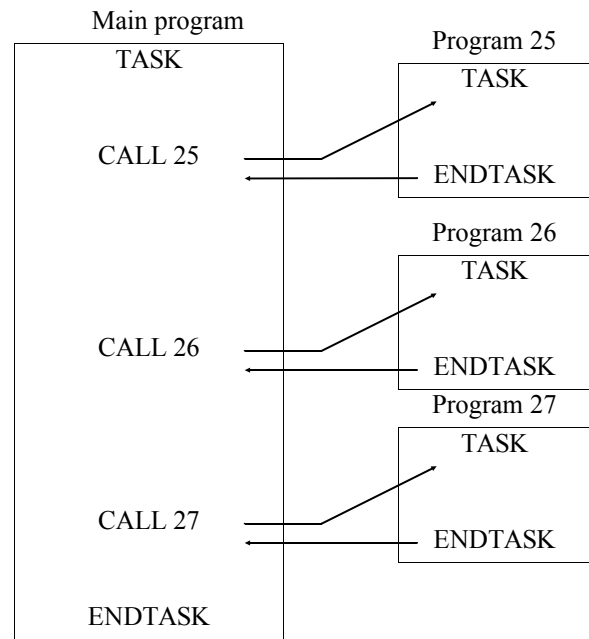


Figure 2: Simple CALL instruction

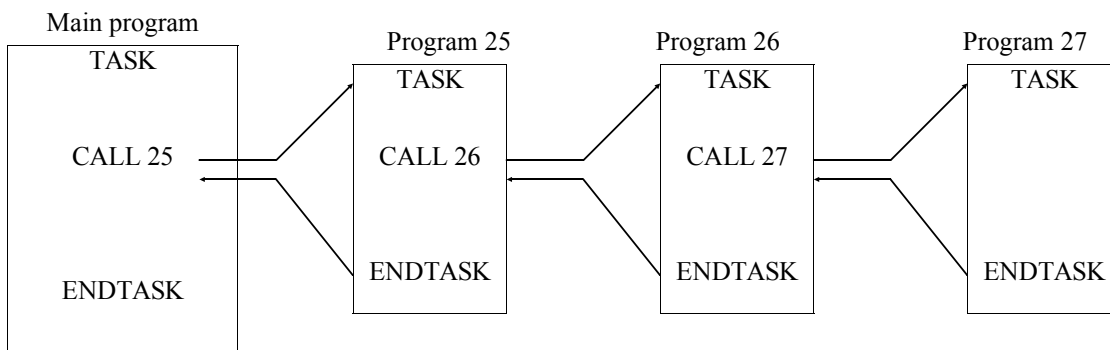


Figure 3: Nested CALL instructions

SEQUENCE CONTROL WITHIN PROGRAMS

The execution sequence of a program can be altered with the instructions GOTO and GOSUB so that portions of the program can be used repeatedly, or only if problems arise.

GOTO

This instruction causes the CRG to search for a label (sequence number) and continue program execution at that point in the program. (1)

usual format: GOTO nn

Where nn is a sequence number in the program.

alternate format: GOTO <variable> (label1, label2, ...)

Where the value of <variable> determines which of the labels control is passed to.

If the value of <variable> is zero or greater than the number of labels then the GOTO is ignored.

EXAMPLE:

```

                                GOTO VI[2] (20, 30, 40)
10    SPEED 20

```

Value of VI[2]	Sequence number control is passed to
0	10
1	20
2	30
3	40
>3	10

Figure 4: Computed GOTO

1 If the specified sequence number is not found in the active program, error 60 will occur at the CRG.

GOSUB

This instruction causes the CRG to search for a label (sequence number) and continue program execution at that point in the program, until a RETURN is found. Then, program execution resumes at the instruction following the GOSUB.

format: GOSUB nn

Where nn is a sequence number in the program. If the sequence number is not found, error 60 will occur at the CRG.

RETURN

Used to end a sequence of instructions which are called by GOSUB. For examples of GOSUB and RETURN see the program listings in the Optimo Software Manual.

IF

This instruction is used to control program execution, by directing flow to a label (sequence number) elsewhere within the program.

format IF <expression> THEN <label> [ELSE <label>]

Where <expression> can be a single variable or any valid logical or mathematical expression. ELSE is optional.

EXAMPLE: IF IL[10] THEN 50

Where control is passed to sequence number 50 if IL[10] is ON.

STOP and PAUSE

These two instructions are only useable while in programming mode from the handbox. Both commands will cause program execution to stop. Pressing the START key on the handbox will continue execution of the suspended part program. In CONSOLE or GATES mode the program will stop, but cannot be re-started.

PROGRAM OFFSET (SYSDEF INSTRUCTION)

The SYSDEF instruction creates a program offset. It is a way to "move" a part program within the machine envelope so that its location corresponds with the actual location of the part's fixture.

The offset uses 6 values, which define 3 translations and 3 rotations. The translations are along the X,Y, and Z axes, respectively. The rotations are around the Z, Y, and X axes.

For example:

```
SYSDEF (50.8, 101.6, 0, 0, 0, 0)
```

moves the part program 50.8 mm (2") along the X-axis, and 101.6 mm (4") along Y axis. Its height (Z-axis location) is not changed.

The SYSDEF instruction follows these rules:

1. Only one SYSDEF instruction may be active; subsequent SYSDEFs simply replace the original.
2. The parameters are real numbers, which can be constants, real variables, or analog I/O.

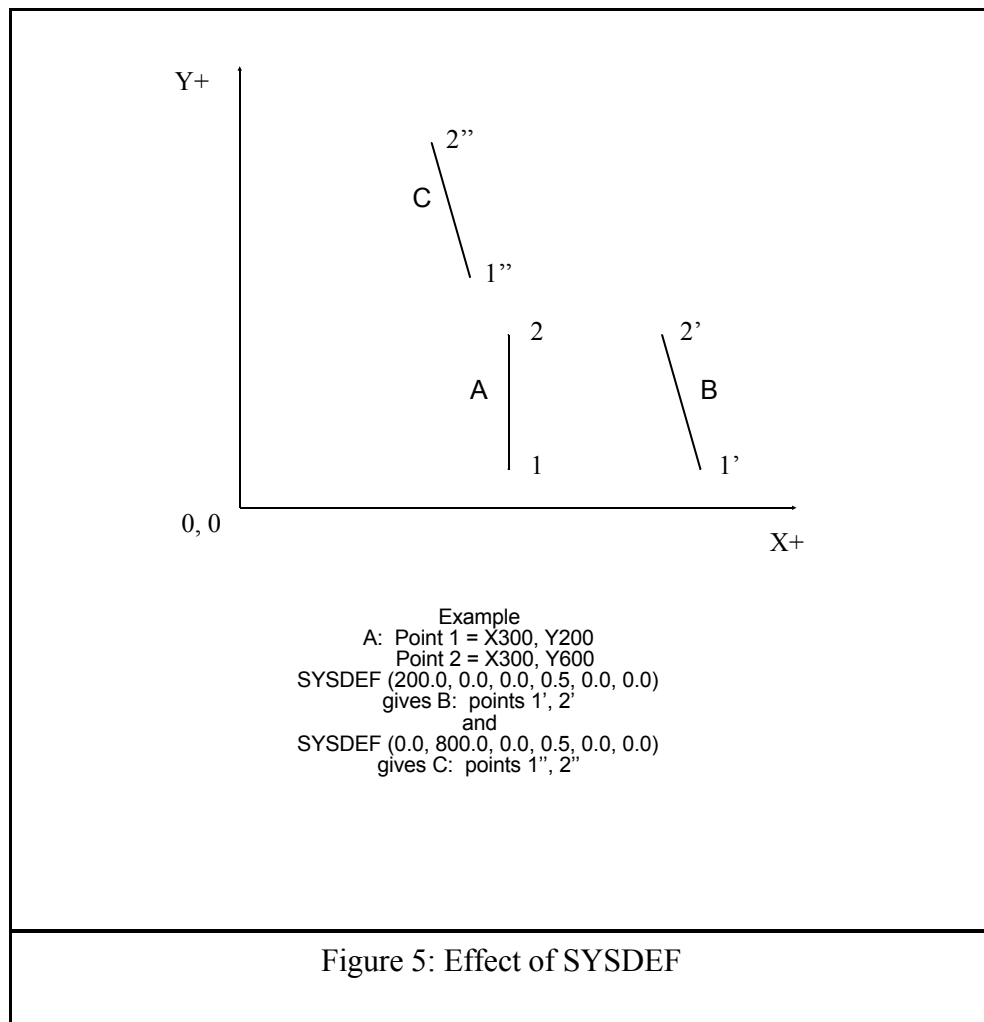
NOTES:

- a. The SYSDEF instruction only affects motion in COORD ABSOL. The user must not use ROBOT mode within a part program except for (if necessary) the initial approach and final retract move(s).
- b. Failure to cancel SYSDEF before teach-mode operations will cause unpredictable results.

The SYSDEF instruction creates a new coordinate system which may be rotated with respect to the basic machine. The rotary axes, however, still refer to the machine coordinate system, not the offset system. This means that the nozzle orientation will change when the part-program is rotated. If a substantial rotation is needed, the off-line transformation program TRANSFOR should be used

- c. If SYSDEF is used in any programs, then all programs should have SYSDEF. If no offset is desired, use SYSDEF without parameters (2) or have a program in the control with the above in it, and use CALL to execute it from your part-program when "zero offset" is desired.

- 2 NOTE: Please do not use SYSDEF with all zeroes "SYSDEF (0.0, 0.0, 0.0, 0.0, 0.0, 0.0)"



DECLARATIONS

In the RML language, "names" can be defined for any constant or variable, using DECLARE.

A name is a string of alphanumeric characters that has between 1 and 8 characters. Names that have more than 8 characters will be cut at the eighth character. Each name must begin with a letter, and cannot be a reserved language word.

For example:

```
DECLARE YHOME = 45.0
```

This gives YHOME the value 45.0, making it a CONSTANT of the numeric type REAL.

"YHOME" can then be used in the program whenever the value 45.0 is needed.

NOTE: YHOME becomes a CONSTANT, not a VARIABLE. If the value must be changed within a part-program, use a variable, as follows:

```
DECLARE CAXIS = OL [11]
```

```
CAXIS = ON
```

will identify the logical output 11 (C-axis sensor) with the name "CAXIS", and turn it on.

Judicious use of the DECLARE directive makes programs much easier to read, edit, and maintain.

The DECLARE directive uses the following rules:

1. A name may be used in only one DECLARE statement per program and may not be duplicated in another DECLARE statement.
2. The DECLARE directives may appear at any point in the program as long as the declaration is made before the name is used. It is recommended, however, that DECLAREs be placed at the beginning of the program.
3. The DECLARE directive becomes part of the part-program, and stays in a program even after it has been compiled and de-compiled.
4. The DECLARE directive does not count as a "program instruction step", does not increment the Handbox step-counter, and is not displayed on the handbox.

MATHEMATICAL AND LOGICAL OPERATIONS

Operations are of two general types; mathematical, and logical. Mathematical operations use numeric values, while logical operations use only ON/OFF (binary) conditions.

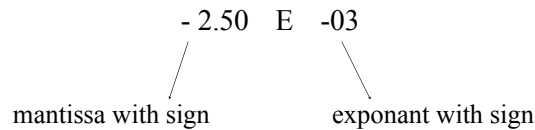
Types of data used in RML

Several types of information is used within part-programs. "Constants" are pieces of information that do not change, once defined. "Variables" can be thought of as storage locations which can hold information, and can change values within the part-program. The Input and Output (I/O) variables are like regular variables, except that some of them are (or can be) "connected" to real, physical, features of the machine.

CONSTANTS

1. A mathematical constant is any number. A constant is considered "integer" if it is between -32768 and +32767, and does not have a decimal point and/or the exponent symbol E.
2. A "real" constant is a number with a decimal point; the exponential format is allowed consisting of sign (+,-), mantissa, "E", sign, exponent

EXPONENTIAL FORMAT



$$\text{value: (mantissa) * (10^{(\text{exponent})}) = -2.5 * 10^{(-3)} = -2.5 * 0.001 = -0.0025$$

Examples of floating-point numbers: 1234.5 .256 E -3 -.5 E +10

The following rules apply:

- The mantissa has 7 digits maximum
- The sign may be omitted if positive
- The exponent must be between 0 and 99

3. PI is a predefined REAL constant equal to 3.14159

4. A logical constant can be the number 0 or the number 1.

The predefined symbols ON, TRUE, and YES have values equal to 1, and the symbols OFF, FALSE, and NO have values equal to 0. These are also valid logical constants.

LOGICAL VARIABLES

LOGICAL variables can only have binary values (one or zero).

The logical type variables are:

- | | | |
|----|------------------|---------|
| 1. | logic input | IL [xx] |
| 2. | logic output | OL [xx] |
| 3. | logical variable | VL [xx] |

where xx is a number between 00 and 99.

NOTE: Logic outputs are used to send commands to the machine from the part-program, and logic inputs are used to read information about the machine into the part-program.

The following is a listing of some commonly-used logic inputs and outputs. For further information, see the section on variable assignments.

LOGIC INPUTS: FUNCTION

IL[16] 1 = START pushbutton is active

LOGIC OUTPUTS FUNCTION

OL[8] = 1 Assist gas 2 select

OL[9] = 1 Assist gas 3 select (option)

OL[11] = 1 Sensor ON

OL[16] For customer use: controls PLC output O97 in PLC cabinet

NUMERIC VARIABLES

INTEGER variables can have values between -32768 and +32767; They can be used as counters, code indicators, pointers etc.

The designation for an Integer type variable is:

VI [xx]

with xx between 0 and 99

No inputs or outputs exist of the Integer type.

REAL variables can have a maximum value of $\pm 2^{63}$ (approx. $\pm 10^{19}$), and a minimum value of $\pm 2^{-64}$ (approx. $\pm 10^{-20}$).

The real type variables are:

1. analog input image IA [xx]
2. analog output image OA [xx]
3. real variable VR [xx]

where xx is a number between 00 and 99.

The following ANALOG INPUTS are predefined

IA[0] .. IA[4] Machine coordinates of Optimo

IA[5] ... IA[39] Reserved

Note that these are "IMAGES", as the system does not have any actual analog input or output that the part-program can access.

EXPRESSIONS

Expressions are used in assigning and in conditional programming. An expression relates two or more data items, using functions or operators. Variables and operations must be of the same "type". (logical operations must use logical variables, and mathematical operations use mathematical variables)

When integer variables and real variables are mixed, the integers are transformed into reals before the operations are performed.

Some examples of legal expressions follow:

OL [15] = IL [32] AND IL [27] [logical]

VR[0] = SIN (VR [5]) +8 [mathematical]

Mathematical Expressions

Mathematical expressions can use any function or operator to link real and/or integer variables and/or constants.

MATHEMATICAL OPERATORS	
+	addition
-	subtraction
*	multiplication
/	division
^	raised to power

Figure 6: Math operators

RELATIONAL OPERATORS	
<	less than
<=	less than or equal to
=	equal to
>=	greater than or equal to
>	greater than
<> (or) ><	not equal to

Figure 7: Relational operators

Parentheses "(" and ")" can be used to change the priority in which the operations are executed. Priority of the elementary mathematical operations, from highest to lowest priority, is as follows:

1. minus-sign (plus sign {+} is ignored).
2. numerical functions..
3. ^.
4. *, /, MOD.
5. +.

When priority is equal, the elementary operation that is located furthest left is executed first.

NUMERIC FUNCTIONS

- | | |
|-------------|---|
| 1. SQRT (x) | square root of (x) [x >0] |
| 2. SIN (x) | sine of x |
| 3. COS (x) | cosine of x |
| 4. TAN (x) | tangent of x |
| 5. ASIN (x) | arc sin. (x) (-1 <=x <=1) |
| 6. ACOS (x) | arc cos. (x) (-1 <=x <=1) |
| 7. ATAN (x) | arc tan. (x) |
| 8. LN (x) | natural logarithm (x) (x >=0) |
| 9. LOG (x) | decimal logarithm (x) (x >=0) |
| 10. EXP (x) | e raised to (x) (-1.0*2**5 <=x <=+1.0*2**5) |

Logical Operations

Logical operations are used to compare on/off conditions, and to set flags or turn logical outputs ON or OFF.

The logical operators are:

- | | | |
|----|-----|-------------------------|
| 1. | OR | logical sum (inclusive) |
| 2. | AND | logical product |
| 3. | XOR | logical sum (exclusive) |
| 4. | NOT | negation |

For logical operations, the order of priority follows (from highest to lowest priority).

1. Operations in parentheses (starting from the most internal ones).
2. NOT.
3. AND.
4. OR and XOR.

When the priority is equal, the elementary operation that is located furthest left is executed first.

ASSIGNMENTS

An "assignment" operation evaluates an expression, and places the result in a variable.

LET is an optional keyword for the assignment operation.

For example:

```
LET  VR [12] = -7.52      using optional LET statement
      VR [12] = -7.52      same effect without using LET
```

The above operation assigns the value -7.52 to the real variable 12.

```
LET  OL [16] = IL [16] or IL [23]      using optional LET statement
      OL [16] = IL [16] or IL [23]      same effect without using LET
```

The example above assigns the value that results from the logical sum of inputs 16 and 23 to output 16.

When assigning values, the use of mnemonic codes (DECLARED names) is allowed, and the word LET is optional. The compiler recognizes everything that does not begin with a recognized keyword as an assignment instruction.

For example:

```
DECLARE X = VR[0]          ; creating name and assigning it to variable
DECLARE XHOME = 10.5      ; creating named constant
X = XHOME + 200.0         ; assigning computed value to named variable
```

Assignment instructions may appear anywhere within a program.

Caution is required when mixing "TYPES" of variables and operations. Unexpected results may occur.

The following rules apply:

1. If the variable in the first part of the statement is logical, the operation in the second part must be logical;
2. If the variable in the first part of the statement is mathematic, the operation in the second part of the statement must be mathematic. If any operands in the second part are integer, they will be converted to reals. If the result is assigned to an integer variable, the result will be converted into an integer by truncation;
3. If the term in the first part of the statement is a logic or an analog output, the result of the assignment will be placed in the corresponding image. The corresponding physical output will be updated to the image value if UPDATE is active;
4. If one or more operands in the second part of the statement are logic or analog inputs the corresponding physical input will first be read and the image in the memory updated. The image will be used for the calculations;
5. The inputs and outputs may be simulated. In this case the calculations are only carried out on the images.

Usage of RML Variables

Certain variables are reserved for AMADA programs and macros. Others are used by the CNC. Of the variables available for customer use, certain divisions are made for purposes of Inch/Metric conversion. If the Inch-Metric conversion module is included in your system, then be careful to not mix "linear" and "rotary" variables, or you may have "unexpected results". See chapter 8 for further information on unit conversions, etc.

The following section lists AMADA's recommendations for variables usage. Failure to follow these recommendations may result in conflicts between AMADA special programs and customer programs.

Assignment of REAL variables

Variable:	Purpose:	
VR 0-39	For customer use in programs- coordinates, etc.	
0-25	These (VR[0] - VR[25]) have linear/rotary assignments	
26-42	These (VR[26] - VR[42]) are not altered by CONVERT.	
They are suitable for passing values to the optional Rotary Table System (FUNCTION 99) and specifying laser Schedules in Workdef 1,nn and Workdef 2,nn		
VR 43-59	Reserved by AMADA (programs 1-7, etc)	
VR 60-97	Customer use: suggested assignments follow	
VR 60-63	work 1,00	Note that the variables VR[60] - VR[79] are considered to have no dimensions, and are not altered by CONVERT.
VR 64 speed 1		
VR 65-68	work 1,01	
VR 69 speed 2		
VR 70-73	work 1,02	
VR 74 speed 3		
VR 75-78	work 1,03	
VR 79 speed 4		
VR 80-85	TCP (remember that the current TCP should also be kept in program 90)	
VR 86-91	SYSDEF 1 (used by optional RMLSHIFT)	
VR 92-97	SYSDEF 2	
VR 98-99	Reserved by the system.	

Assignments of INTEGER variables

VI [1]	program number (for chaining)
VI [2] - [69]	open -for program counters,etc
VI [70] - [97]	reserved by AMADA
VI [98] - [99]	used by system

Assignments of LOGICAL variables

VL [0] - [79]	open
VL [80] - [99]	reserved by AMADA

Assignments of Program I/O

OL[0] - OL[1]	used by system
OL[2]	Modulation on/off (EFA-51 systems only) other systems: used
OL[3] - OL[7]	used by system
OL[8]	Cutting gas 2
OL[9]	Cutting gas 3 (option) (3) • NOTE: only one assist gas may be selected at one time.
OL[10]	used by system
OL[11]	C-axis sensor (ON/OFF)
OL[12] - OL[15]	used by system
OL[16]	For customer use: controls the PLC output O97 (available in PLC cabinet)
OL[17] - OL[31]	used by system
IL[0] - IL[15]	used by system
IL[17] - IL[31]	used by system

- 3 If system has only two cutting gas regulator/solenoids available, turning OL[9] ON results in System Alarm 931

READ ONLY variables

Variable	purpose
IL[16}	Status of START push-button: Can use in program as MOVE (...) / START (IL[16])
IL 87-94	work flag process 0-7 (1=work ON)
IL 95	motion flag OPTIMO (1=moving)
IL 96-99	reserved
IA 00-04	coordinates of OPTIMO X = IA[0] Y = IA[1] Z = IA[2] A = IA[3] B = IA[4]
	<ul style="list-style-type: none"> • Tool tip position values (X, Y, Z) will be returned only if COORD ABSOL is active; otherwise you get ROBOT values for X, Y, Z, which are for the Z-axis column at the intersection of A-Axis and B-axis centerlines. Head angles are the same in either ROBOT or ABSOL.
IA 09-35	reserved
OA 0	speed (mm/sec * 48ms) (4)
VI 98	execution counter for current program
VI 99	most recent error code

- 4 The value of (48 ms) or (64 ms) depends on system hardware and firmware. The active system value may be determined from system calibrations. See the Service Manual or call AMADA for more information.

EXAMPLES OF USE OF VARIABLES

(calling program, metric format)

```

; sysdef variables
VR[86] = 0.          ; table offset X - for optional RMLSHIFT
VR[87] = 0.          ; table offset Y          "
VR[88] = 0.          ; table offset Z          "
; tcp variables
VR[80] = 0
VR[81] = 0
VR[82] = 5          ; shifting the tip 5 mm from reference location
VR[83] = 0
VR[84] = 0
VR[85] = 0
; speed variables
VR[64] = 5.5
VR[69] = 3.2
; workdef variables
VR[65] = 4.5
VR[66] = 95
VR[67] = 1.2
VR[68] = 99
CALL 45             ; target program 1
VR[86] = 30.        ; second program X30.0 mm from its programmed location
CALL 46             ; target program 2
CALL 2              ; Z-UP and cleanup at program end
(target program 1, CRG program #45)
SYSDEF ( VR[86], VR[87], VR[88], 0., 0., 0.)
TCP ( VR[80], VR[81], VR[82] / VR[83], VR[84], VR[85] )
SPEED 30
WORKDEF 1,01 ( VR[65], VR[66], VR[67], VR[68] )
. (program data here)
WORK 1,01/ON
SPEED VR[64]
(program data here)

```

OFF-LINE CONVERSIONS

CUTTER COMPENSATION

Cutter Compensation is an optional package for use with the Optimo system. It permits offsetting the tool path a programmed amount to correct for laser kerf width. Cutter Compensation is implemented as an operation "off-line", on the MMI computer. The source program is read, and a new program is created with new coordinates (where necessary) incorporating the offset directions and amounts.

While the programmed offset amounts are converted between Inch and Metric whenever the source program is converted, the cutter compensation may only be performed when the source file is in METRIC format.

INSTRUCTIONS

Cutter comp instructions are provided for distance and direction of offset. Each instruction must be preceded with a special marker (;\$) to identify it to the processing software. The results of cutter comp are dependant on these cutter comp instructions and on the part-program format and contents.

Instructions are not case-sensitive; they may be any combination of upper-case and lower-case desired.

OFFSET AMOUNT

;\$dia specifies diameter of laser beam to use for offset:
offset amount will be half of beam diameter.

;\$rad specifies radius of laser beam to use.
This value is used for offset distance.

OFFSET DIRECTION

;\$left specifies that the new program path will be to the left of the original path.

;\$right specifies that the new program path will be to the right of the original path.

;\$none (default) specifies that no compensation will occur

OFFSET HANDLING

The Cutter Comp software assumes that the cutting head is always kept perpendicular to the part surface. It then "looks at" the coordinates of each MOVE instruction to determine direction of cut. Offset is computed looking at the surface of the part and direction of cut, to determine "left" and "right" of path.

If the cutting tip is at an angle to the workpiece, the offset will not be parallel to the actual part surface.

For PATH HOLE, cutter comp is performed according to the convention that the hole is contoured clockwise "looking into the axis". Therefore, ;\$CLEFT makes the hole LARGER and ;\$CRIGHT makes the hole SMALLER.

OUTPUT FORMAT

The output program is formatted identically to the input program, except for numeric values in MOVE instructions where cutter comp is active. Those numeric values are formatted to five decimal places.

This can be used to determine if cutter comp was active in a particular section of a part-program.

BEGIN COMP

Compensation starts when an offset amount and direction have been specified, a contouring PATH type has been selected, and the laser beam is turned on.

PATH PATH PTP is assumed to be the machine default. No compensation is performed until a va instruction for LIN, CIR, or HOLE.

WORK no compensation occurs until a WORK n,nn/ON is encountered in the program.

HALT COMP

Compensation set to zero after WORK n,nn/OFF is encountered in the part-program.

CHANGE COMP

The direction and/or distance of offset can be changed only when cutter comp is inactive (laser beam is OFF)

INCH/METRIC CONVERSION

The Optimo laser machine is programmed using source files written in the Robot Machine Language (RML). When an RML program is prepared for the machine, all linear units must be specified in millimeters and all rotary units must be specified in radians. The Optimo MMI system uses a software package (CONVERT) to permit programming in the inch-degree system of units. When a part-program on the MMI is prepared for the machine, CONVERT is run to perform necessary unit conversions. The reverse process is available when a file is transferred from the machine to the MMI, so that the user can edit and program (off-line) in whichever unit system is preferred.

The MMI system assumes that a part-program's file extension describes its unit system. A MM-RADIAN program will be named NAME.S", while an INCH-DEGREE program will be named "NAME.SI". (the MMI / Scheduler uses other file extensions as well- see the section on filenames and conventions)

When part-programs are moved between the MMI computer and the CRG, necessary conversions are performed automatically. For program test purposes, the operation may also be performed manually.

The input to the CONVERT program is a legal RML program. The output is an equivalent legal program in which all numbers used as linear and rotary measurements are converted. Only literals (actual numbers) are affected by CONVERT.

Literals with linear units are converted from inch to mm or vice-versa. Literals with rotary units are converted from degrees to radians or vice-versa. Literals which are not linear or rotary measurements are not changed by CONVERT. Similarly, words and symbols are not affected.

Values associated with the optional Cutter Comp package are also handled when conversions are performed. ;\$RAD and ;\$DIA are considered to have linear units.

When an inch/degree program is to be compiled, the program must be converted from inch/degree to mm/radian format. When a mm/radian program has been decompiled, the program may be converted to inch/degree format. CONVERT may be used for these operations.

Figure 1 shows how CONVERT fits into the development system.

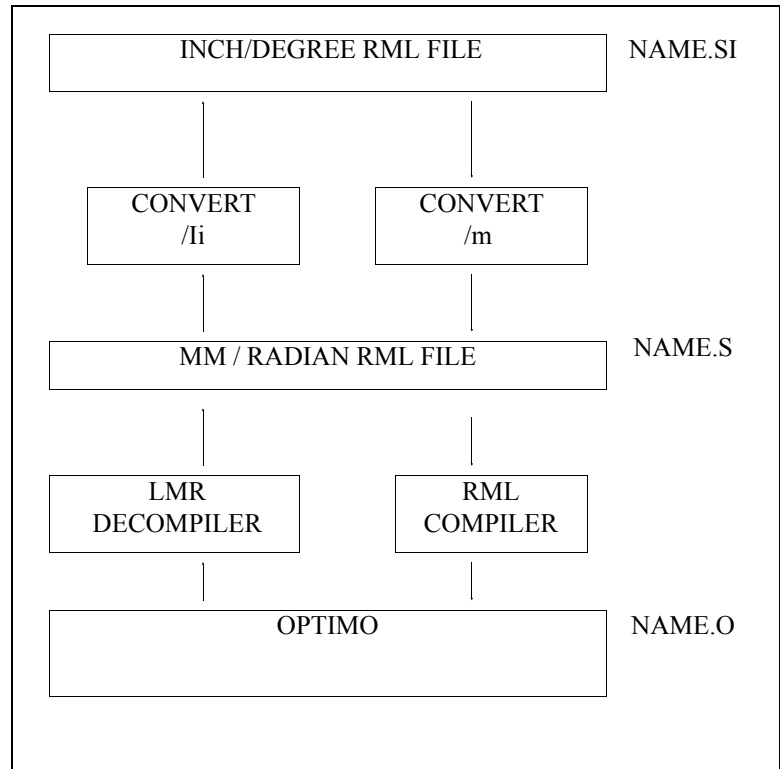


Figure 1: File conversion

Format of Examples

For clarity, examples in the text will have the units and other pertinent information noted after a semicolon. These units and comments do not appear in the actual RML program. Equivalent statements (before and after being converted), are shown in the following format:

FORMAT

```
input instruction ;degrees
```



```
output instruction ; radians
```

EXAMPLE

```
DECLARE XDIST = VR[5] ;XDIST is linear
```

```
XDIST = 230 ;inches
```



```
DECLARE XDIST = VR[5]
```

```
XDIST = 5842 ;mm
```

Figure 2: Format of examples

PREDEFINED VARIABLES

Some of the VR variables are predefined to be either linear or rotary variables. Literals associated with these variables will be considered to have the same units.

The following are linear variables:

- VR[0]..VR[2]
- VR[5]..VR[7]
- VR[10]..VR[12]
- VR[15]..VR[17]
- VR[20]..VR[22]
- VR[80]..VR[88]
- VR[92]..VR[94]

The following are rotary variables:

- VR[3]..VR[4]
- VR[8]..VR[9]
- VR[13]..VR[14]
- VR[18]..VR[19]
- VR[23]..VR[24]
- VR[89]..VR[91]
- VR[95]..VR[97]

All VR variables not listed above do not have any predefined units. Literals associated with these variables will not be converted.

ROTARY TABLE

The P&W Rotary Table is always programmed in DEGREES. Do not use a defined rotary variable in FUNCTION 99, as it would be converted to RADIANS and cause improper operation.

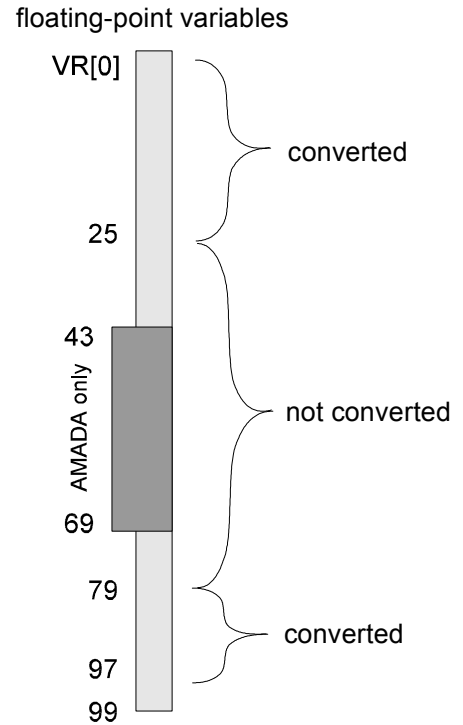


Figure 3: REAL variables

SYMBOL DECLARATIONS

RML allows the programmer to declare symbolic names for variables and constants. The DECLARE instruction takes the following form.

DECLARE symbol = variable

or

DECLARE symbol = literal

In the case of variables, if the variable in the declare instruction is a VR variable, then CONVERT considers the new symbol to have the same units as the VR[n] variable.

- Example:

DECLARE YNEAR = VR[15]

YNEAR would be considered a linear variable, because VR[15] is a linear variable.

- Example:

DECLARE BPOS = VR[3]

BPOS would be considered a rotary variable, because VR[3] is a rotary variable.

If the variable in the DECLARE statement is not a VR variable with assigned units, then the symbol is considered neither linear or rotary. It is ignored by the CONVERT program.

When the DECLARE statement contains a literal, instead of a variable, CONVERT cannot determine what the units are. Therefore, the literal is not converted.

- Example:

DECLARE EPSILON = 0.001

The literal '0.001' will not be converted.

CONVERSION PROCESS

As CONVERT processes the input program, it analyzes each instruction and decides if any literals it contains should be converted. The following sections discuss the various instructions which may contain linear or rotary literals that need to be converted.

ASSIGNMENTS

Assignment instructions allow the programmer to give a symbol or variable a specific value. This instruction has the following form.

LET symbol = value

The word 'LET' may be omitted.

If the value in the LET instruction is a literal, CONVERT considers it to have the same units as the symbol .

Example:

LET VR[3] = 1.5 ;degrees

^

||

v

LET VR[3] = 0.026175 ;radians

The literal '1.5' will be converted in rotary units because VR[3] is a rotary variable.

Example:

DECLARE YNEAR = VR[15] ;YNEAR is linear

YNEAR = 230 ;inches

^

||

v

DECLARE YNEAR = VR[15]

YNEAR = 5842 ;mm

The literal '230' will be converted in linear units, because YNEAR has been declared as a linear variable.

The right hand side may be an expression. This subject is discussed in section 4.3.

INSTRUCTION PARAMETERS

Some RML instructions require the programmer to provide information in the form of parameters. For example, the MOVE instruction has five parameters indicating a position coordinate. Parameters may be passed in the form of literals, symbols and expressions. Literals and symbols are discussed in this section. Expressions are discussed in the next section.

Parameters may have specific units (linear, rotary or neither). CONVERT assumes that the actual parameter values, whether in the form of literals, symbols, or expressions, are in the specified units.

LINEAR AND ROTARY PARAMETERS

The following instructions have linear or rotary parameters. The following list illustrates the required units. Linear parameters are represented by the word LIN. Rotary parameters are represented by the word ROT.

MOVE (LIN, LIN, LIN, ROT, ROT) / VIA (LIN, LIN, LIN, ROT, ROT)

SYSDEF (LIN, LIN, LIN, ROT, ROT, ROT)

TCP (LIN, LIN, LIN/ LIN, LIN, LIN)

SIN (ROT)

COS (ROT)

TAN (ROT)

- Example:

MOVE (1347.5, 1001.0, -241.2, 12.54, -1.42) ;mm/radians

^

||

v

MOVE (53.05, 39.41, -9.5, 718.62, -81.38) ;inch/degrees

The first three parameters in the MOVE instruction are linear, the last two are rotary.

- Example:

TCP (8.68, 0, 0.72/0, 1.77, 0) ;inches

^

||

v

TCP (220.5, 0, 18.3/0, 45, 0) ;mm

All parameters in the TCP instruction are linear.

OTHER PARAMETERS

All parameters and variables not listed above will be considered unitless, and any associated literals will not be converted. For example, literals used as parameters in SQRT, LN, LOG, and EXP instructions will not be converted.

- Example:

$$\text{LEN} = \text{SQRT}(2)$$

The literal '2' is not converted.

EXPRESSIONS

RML allows arithmetic, logical, or relational expressions to be used wherever numerical or logical data is needed.

Arithmetic expressions contain the operators: +, -, *, /, and ^.

Relational expressions contain the operators: <, <=, >=, >, =, and <>.

Logical operators are: OR, AND, XOR, and NOT.

All of the above operators work with two operands (literals, symbols, or expressions). In the case of addition and subtraction, and all the relational operators, CONVERT assumes that both operands have the same units.

Only one operator is allowed in each expression. If more than one operator is present, the numbers in the expression are not converted.

- Example:

$$\text{VR}[3] = \text{VR}[3] + 180 \quad ;\text{degrees}$$

$$\wedge$$

$$\parallel$$

$$\vee$$

$$\text{VR}[3] = \text{VR}[3] + 3.141 \quad ;\text{radians}$$

The literal '180' is converted as a rotary number, because VR[3] is a rotary variable.

- Example:

```
Y = YNEAR - 25.4 ;mm
      ^ ;YNEAR is linear.
      ||
      v
```

```
Y = YNEAR - 1.0 ;inches
```

- Example:

```
IF DIST 100 THEN 50 ;inches
      ^ ;DIST is linear.
      ||
      v
```

```
IF DIST 2540 THEN 50 ;mm
```

- Example:

```
/START (BPOS 6.182) ;radians
      ^ ;BPOS is rotary.
      ||
      v
```

```
/START (BPOS 360) ;degrees
```

However, operands for the *, /, and ^ operators do not necessarily have the same units. Therefore, literals which are divided, multiplied, or raised to the power, are not converted by CONVERT.

- Example:

```
DIST = NEAR * 2
```

The literal '2' will not be converted, because its units cannot be determined.

Logical operators work only on logical operands (true, false). Therefore, CONVERT ignores them.

CONVERSION RATIOS

Convert uses the following conversion ratios:

$$1 \text{ inch} = 25.4 \text{ mm}$$

$$1 \text{ mm} = (1 / 25.4) \text{ inches}$$

$$1 \text{ degree} = ((2 * \text{PI}) / 360) \text{ radians}$$

$$1 \text{ radian} = (360 / (2 * \text{PI})) \text{ degrees}$$

When calculated, these numbers will have an accuracy of about fifteen digits in decimal. Because the memory on a computer is fixed and finite, some accuracy may be lost when real numbers are divided and multiplied. Therefore, the value of a number may change slightly if converted many times. The programmer should keep this in mind and check his literals occasionally if he converts them and reconverts them many times consecutively.

USING CONVERT FROM DOS

The program is run by typing the word 'CONVERT', the name of the input file, an optional output filename, and the appropriate optional switch. Typing just the word 'CONVERT' will display the parameter explanations on the CRT monitor.

When an inch/degree program is to be compiled, the program should be converted from inch/degree to mm/radian format. In this case, CONVERT is run with the /m switch. The switch may be omitted if the RML program filename has the extension .SI. The output program filename will have the default extension of .S. The output file is then compiled.

When a mm/radian program has been decompiled, the program should be converted to inch/degree format. In this case, CONVERT is run with the /i switch. The switch may be omitted if the RML program filename has the extension .S. The output program filename will have the default extension of .SI. The output file can then be edited.

Convert has the following basic form:

CONVERT filename [.ext] [out] [switches]

filename

The name of the file to be converted.

[.ext]

An optional filename extension. If the extension given is ".S", Convert automatically knows that it must convert a millimeter/radian program to an inch/degree program. If the extension given is ".SI", Convert automatically knows that it must convert an inch/degree program to a millimeter/radian program.

[out]

An optional output filename. If out does not include an extension, then CONVERT will add the default extension. If the output filename is omitted, then the new file containing the converted numbers will have the same name as the original program except that it will have the default extension. The default extension is .SI if the original program was in millimeter/radian format. The default extension is .S if the original program was in inch/degree format.

[switches]

/i

Tells Convert that it must convert a millimeter/radian program to an inch/degree program. If this switch is specified when the file extension given is ".SI" (already an inch/degree program), an error will be displayed and the conversion will not take place.

/m

Tells Convert that it must convert an inch/degree program to a millimeter/radian program. If this switch is specified when the file extension given is ".S" (already a millimeter/radian program), an error will be displayed and the conversion will not take place.

- Examples:

COMMAND LINE DEFAULT OUTPUT FILENAME OUTPUT FORMAT

CONVERT Displays parameter explanations on the CRT.

CONVERT part1.s part1.si inch/degree

CONVERT newbit.si newbit.s mm/radian

CONVERT optimo.old /i optimo.si inch/degree

CONVERT optimo.new /m optimo.s mm/radian

CONVERT part1.s /m Error condition.

CONVERT newbit.si /i Error condition.

ERROR MESSAGES

The following error messages are used by CONVERT. They will only be seen if running CONVERT from DOS, as otherwise Scheduler handles parameters, etc.

FATAL ERROR EXPLANATION

Conflict With Filenames and Switches

The extension(s) on the filename(s) conflict with each other or with the specified switches. CONVERT cannot decide whether to convert to inch or metric.

Illegal Switch

One or more of the switches specified is illegal.

Input File Could Not be Opened

The input file specified does not exist, or cannot be read.

Must Specify Conversion Required

Either the input file must have an extension of .S or .SI, or a switch must be specified. CONVERT cannot decide whether to convert to inch or metric.

Not Enough Arguments

At least one filename must be specified.

Output File Could Not be Opened

The filename specified for the output file is illegal.

Too Many Arguments

Only two filenames, input file and output file, can be specified.

Too Many Symbols in Line

The instruction is too complex for CONVERT to handle. The last part of the line may not be printed out.

ADDITIONAL INFORMATION

This chapter covers additional features of the RML language and its use on OPTIMO. The features include the FUNCTION instruction and others.

FUNCTION instruction.

FUNCTION may be used to call other programs or modules as needed. It is the method of accessing certain optional devices such as an rotary table or optical probe system. It can also be used to run standard operations such as Z-Up, or user-created programs and utilities.

FUNCTION is a compromise between the CALL and the GOSUB instructions. It is faster than CALL, and has the advantage of creating a single executable file, which may be easily archived. Compared to the GOSUB it has the following advantages:

- 1) The called program can be created and tested independently from the calling one.
- 2) The called program doesn't need to be merged into the calling one at the source level.
- 3) FUNCTION can be programmed in self-teaching mode as one instruction (macro function).

The linkage of calling and called program is accomplished by the compiler during the second pass, the result is a program in executable format (1). At the end of the main program compilation, all the function (programs) are compiled and appended at the end of the main program as routines (each routine has its ENDTASK replaced with a RETURN instruction). The size of the final program must be smaller than 32 Kbyte. (2)

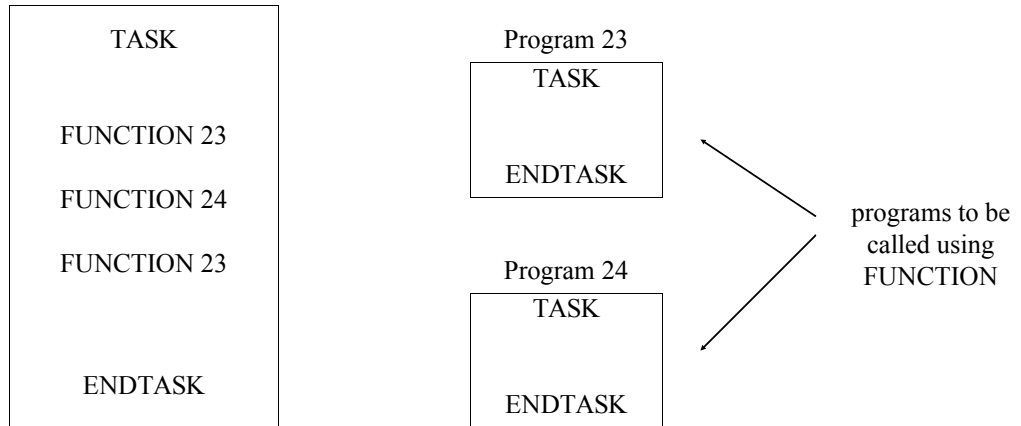
The functioning can be summarized as follow:

- Up to 32 different functions may be in a single executable program.
(Each function is compiled only once)
- The number that identifies the function must be different from the identifier of the main program.
(For example: program 25 can't invoke FUNCTION 25)
- 4 nesting levels are allowed, and a function can call itself (recursion is supported).
- The program implementing a function must define its entry point: this is obtained with the instruction "TASK 1". The program must then have at least one instruction TASK 1 or no instruction TASK at all.
- If a program implementing a function is modified, it is necessary to do the second pass compilation of all programs calling that function.

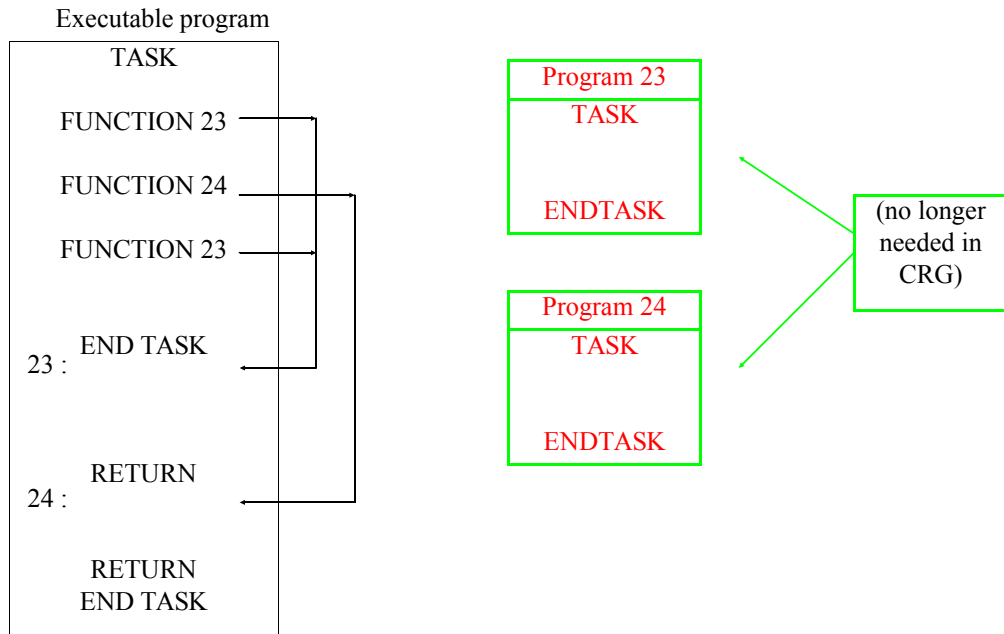
From the CRG, force the creation of a new executable by modifying the main program or deleting its executable files.

- 1 If the second pass compilation is done off-line (using the C2 compiler) then each program named in a FUNCTION instruction must be present on disk where the compiler can find it.
The name of each "function" file must be "n.o", where n is the number of the function as is called from the main program.
- 2 The Object-format files for the various functions must be present when the main program is compiled. The executable files for the individual functions are not needed for compilation or execution.

The source and the object codes of a program containing FUNCTION instruction will use different files, but the executable format will reside in one unique module. (see following illustration)



Composition of source, object files (before phase 2 compilation)



Composition of executable file (created by phase 2 compilation)

Figure 1: Files created using FUNCTION

Example of use

In the following example two parts are similar but not identical. On each part an identical rectangle must be cut. (same size, same position)

The main programs will be designated Nr. 32 and Nr. 68. The rectangle may be taught only once and then merged using the FUNCTION instruction. The rectangle is assigned program nr. 44

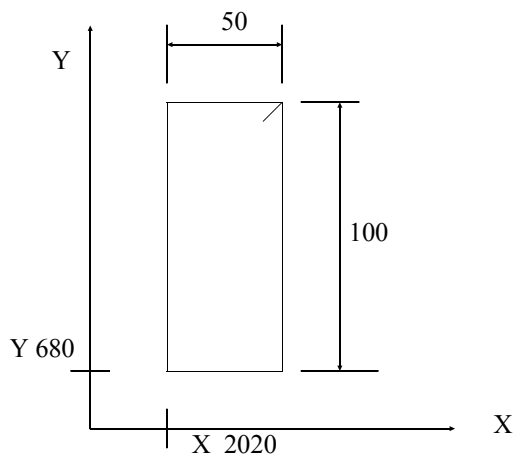
EXAMPLE:

PROGRAM 44 LISTING:

```

PATH LIN
SPEED 30                                ; approach speed
MOVE (2070., 700., -530., 0., 0.) / FLY ; positioning move
OL [11] = ON                             ; sensor on
SPEED 10                                ; approach, cutting speed
                                           ; (note that we're approaching surface at angle)
MOVE (2060., 770., -550., 0., 0.)      ; piercing position
WORK 0, 00 /ON                          ; beam ON
MOVE (2070., 780., -550., 0., 0.)      ; first corner
MOVE (2070., 680., -550., 0., 0.)      ; second
MOVE (2020., 680., -550., 0., 0.)      ; third
MOVE (2020., 780., -550., 0., 0.)      ; back to first
WORK 0,00 / OFF                         ; beam off
SPEED 30
MOVE (2070., 780., -530., 0., 0.)      ; retract
OL [11] = OFF                           ; sensor off
                                           ; end of program 44

```



When programs 32 and 68 are compiled, the code for program 44 will be appended to them.

The main part-programs, nr. 32 and 68 are:

- Program 32

```

COORD ABSOL
PATH . . . . . ; setup, initial cutting
MOVE . . . . . ;
MOVE . . . . . ;
. . . . . ;
FUNCTION 44 ; cut rectangle
MOVE . . . . . ;
MOVE . . . . . ; finish part
MOVE . . . . . ;
MOVE . . . . . ;
. . . . . ;
ENDTASK ; end of program 32

```

- Program 68

```

COORD ABSOL ;
PATH . . . . . ; setup, initial cutting
MOVE . . . . . ;
MOVE . . . . . ;
FUNCTION 44 ; cut rectangle
MOVE . . . . . ;
MOVE . . . . . ; cut rest of part
MOVE . . . . . ;
MOVE . . . . . ;
ENDTASK ; end of program 68

```

The GOSUB and CALL instructions can also be used to accomplish the same job, but GOSUB requires you to manually insert the source code of the feature to use (the RML source code contained in program 44) into each source program which will use it. The CALL instruction permits larger called routines, but takes longer to run. (The overhead of making a CALL occurs each time it is used. This is especially important for an operation to be run many times.)

PASSING PARAMETERS AMONG MODULES.

CALL and FUNCTION allow the exchange of data between calling and called module. The communication is made through "formal parameters" appropriately defined or by means of system global variables (VR, VI, etc...), in this case we have a bidirectional communication.

The maximum allowed number of parameters is 8.

Example of syntax:

```
FUNCTION 6 (VR[40], 5, DATAIL, IL[4])
```

```
CALL 6 (VR[40], 5, DATAIL, IL[4])
```

If the communication is made through formal parameters then you must define the type of parameter with the instruction DEFINE in the called program.

Formal parameters are identified by the following mnemonic names: PAR[0], PAR[1],.... PAR[7]

The allowed types are:

- FLOAT: for PAR of float type (constant, VR, IA, OA)

- INT : for PAR of type integer (constant, VI)

- LOGIC: for PAR of type logic (IL, OL, VL)

The PAR[n] variables are local to the called module; the system variables are global. To pass parameters from the called program to the calling program, use global variables.

The default numeric type of the PAR[n] variables is FLOAT.

Usage EXAMPLE of formal parameters.

```
DEFINE PAR[0] = FLOAT
```

```
DEFINE PAR[1] = LOGIC
```

```
DEFINE PAR[2] = INT
```

```
DECLARE radius = PAR[0]
```

```
LET VR[5] = radius
```

- NOTE:

The compiler does not make any check on the congruency between the type of parameters of the calling program and the formal parameters of the called program. THE PROGRAMMER MUST ASSURE SUCH CONGRUENCY.

EXAMPLE:

```
DEFINE PAR[0] = INT    legal
```

```
DEFINE VR[1] = PAR[0]  illegal
```

In the following example we will pass an integer between two modules using the FUNCTION instruction.

```
DECLARE DATO = VI[30] ; Assign name DATO to the integer variable VI[30]
```

```
FUNCTION 44(DATO)
```

The FUNCTION module will contain:

```
DEFINE PAR = INT ; type declaration
DECLARE DATO1 = PAR[0] ; assign symbolic name
```

```
.
```

```
.
```

Syntax of DEFINE

- DEFINITION:

Defines the formal parameters for the communication between calling and called program.

- SYNTAX:

```
DEFINE (parameter identifier) = (type of value to pass)
```

- PARAMETERS

- (parameter identifier) is PAR[n] where n = 0 - 7

- (type of value to pass) is one of the following reserved symbols

 FLOAT: for PAR of floating-point type (constant,VR,IA,OA)

 INT : for PAR of type INTEGER (constant,VI)

 LOGIC: for PAR of type LOGIC (IL,OL,VL)

- EXAMPLES:

```
DEFINE PAR[0] = FLOAT
```

```
DEFINE PAR[1] = LOGIC
```

```
DEFINE PAR[6] = INT
```

- RELATED INSTRUCTIONS:

```
DECLARE            LET
```

- NOTES:

It is a global type instruction

DEFINE must be used before the reference to the PAR defined.

HANDBOX OPERATIONS

Use of function and debug keys on Handbox

KEY	MODE USED	FUNCTION
F1	MANUAL	Not used on OPTIMO
F2	MANUAL	OVERSH instruction (determines % of operating speed to maintain in a MOVE with the / FLY attribute)
F3	MANUAL	FUNCTION instruction (see text)
SHIFT-F3	MANUAL	SYSDEF instruction
F4	MANUAL	Not used on OPTIMO
F5	MANUAL	Not used on OPTIMO
F6	MANUAL	SIMUL function (dry-run program and begin motion on specified instruction)
F7	MANUAL	Not used on OPTIMO
F8	MANUAL	GOSUB instruction
DEB	MANUAL	BREAK function (program halt after specified program instruction)
PAR	CYCLE REPEAT	Not used on OPTIMO
DISP	CYCLE REPEAT	Not used on OPTIMO

	F1	F2
DEB	F3	F4
PAR	F5	F6
DISP	F7	F8

Figure 2: Handbox function keys

FUNCTION From the Handbox

During self-teaching it may be useful to run / store programs as modules. This operation can be achieved through the FUNCTION instruction. This instruction can be called from the handbox by pressing the F3 key. The following mask is displayed:

FUNCTION = XX

Key-in the program number to use, and press ENT.

NUMBER PARAMETERS = X

Key-in the number of parameters to pass and press ENT. (3)

The display will then prompt for values for the parameters. The PREV and NEXT keys select between the parameters.

PAR0 = XXXX.X

PAR1 = XXXX.X

PARX = XXXX.X

The values shown in the field "XXX.X" are real numbers (4). The ENT key confirms the values and ends the operation.

The next prompt asks you if you want to execute (run) the function.

EXECUTION?

If NO is pressed, then the FUNCTION mask is presented again. Press REC to record the instruction into the part program.

If YES is pressed, then the function will be run. Any axis motion will occur at 40% of the selected handbox feedrate. The STOP and START keys may be used to control execution of the function. To abort the execution of the function, press the SHIFT and STOP keys.

When the function has finished execution, it may be stored in the program by pressing the REC key. The following mask is then displayed.

- 3 From 0-8 parameters may be passed, depending on the program being called. The Rotary Table System (FUNCTION 99) takes one parameter. Programs such as Z-Up, Home Return, etc. do not take any parameters.
- 4 The handbox will show real numbers for each parameter. If the called program expects another numeric format, the operator must enter values which fit the expected format.

P = XX S = YYYY FUNC = NN

Where NN is the function number.

The FUNCTION can be run as a stand-alone program.

SAMPLE PROGRAMS

This chapter has examples of various language features and programming techniques.

SAMPLE 1

The first sample program is fairly simple. It does use SYSDEF, selects a laser schedule, and it uses the sensor and all four types of PATH. Note that the source file is not case-sensitive.

; Metric File

```
1          COORD ABSOL
2          path ptp
3          SPEED 30
4          SYSDEF (1034.7706, 1427.226, -569.722, 0.0, 0.0, 0.0)
5          WORKDEF 2,00 (11)
6          WORK 2,00 /ON
7          WORKDEF 0,0
8          MOVE (108.0, -284.0, 300.0, -2.8, 0.0)/fly
9          SPEED 6
10         MOVE (0.0, 0.0, 1016.0, -2.871643, -1.570796)
11         path hole (12.7)
12         work 0,0/on
13         MOVE (0.0, 0.0, 1016.0, -2.871643, -1.570796)/via(0.0, -254.0, 1016.0, -2.87164
14         work 0,0/off
15         path lin
16         MOVE (0.0, 0.0, 0.0, -2.871643, -1.570796)
17         SPEED 7
18         WORK 0,0 /ON
19         DELAY .5
```

SAMPLE 2

The second sample program is derived from a customer program. This program is for a system with EFA-51 laser, using outputs 12, 13, 14, 15 for laser control.

; Metric File C:\CST\INCH\CYLIND.S Created From Inch File C:\CST\INCH\CYLIND.SI - Thu Mar 04 08:52:34 1993

```
;*          A. REPLACED SYSDEF WITH VARIABLES TO ALLOW OPERATOR
;*          ADJUSTMENT OF THE 2 OPENINGS IN Z INDEPENDENT OF
;*          THE TRIM AND OPENINGS.
```

```
;*          PROGRAMMERS NOTE:
```

```
;*          A. PROGRAMMED X AND Y ZERO ARE THE PART ORIGIN.
;*          Z-ZERO IS THE PART SEATING SURFACE (DATUM -A-).
;*          (DATUM -A- HAS .050" STOCK ON).
```

```
;*          B. VR[10]= SYSDEF X      VR[11]= SYSDEF Y
;*          VR[12]= SYSDEF Z FOR TRIM.
;*          VR[13]= SYSDEF Z FOR LARGE OPENING
;*          VR[14]= SYSDEF Z FOR SMALL OPENING
```

```
*****
```

```
VR[10] = 1262.139005      ; X SYSDEF ENTIRE PROGRAM
VR[11] = 719.939988      ; Y SYSDEF ENTIRE PROGRAM
VR[12] = -832.442988     ; Z SYSDEF TRIM ONLY
VR[13] = -832.697       ; Z SYSDEF LARGE OPENING ONLY
VR[14] = -832.697       ; Z SYSDEF SMALL OPENING ONLY
```

```
SYSDEF (VR[10],VR[11],VR[13],0.0,0.0,0.0)
```

```
; NOTE: THE ABOVE SYSDEF SHOULD ONLY EVER BE CHANGED BY
; INCREMENTS OF 3.5" IN X AND 3.0" IN Y.
```

; THE Z SYSDEF INCLUDED A .500" BLOCK SITTING UNDER
;
; FIXTURE. IF THE .500" GAGE BLOCKS ARE REMOVED
;
; THE Z SYSDEF VALUE HAVE 12.7 MM ADDED (MORE MINUS).

WORKDEF 0,00

OL[13]=ON

OL[8]=ON

COORD ABSOL

PATH LIN

SPEED 33.

; BURN THE -409 BOSS OPENING BVCL...

; A=0, B=PI/2

MOVE (95.35701,-163.994998,420.499997,0.528,1.5708)/FLY

MOVE (95.35701,-163.994998,190.5,0.528,1.5708)/FLY

SPEED 10.

OL[11]=ON

MOVE (70.222008,-120.771996,190.5,0.528,1.5708)

WORK 0,00/ON

DELAY .25

MOVE (70.222008,-120.771996,191.77,0.528,1.5708)/FLY

PATH CIR

MOVE (0.0,-139.703988,262.001,0.0,1.5708)/VIA(49.659007,-130.580003,241.429007,
0.364,1.5708)/FLY

MOVE (-70.217995,-120.774003,191.77,-0.528,1.5708)/
VIA(-49.659007,-130.580003,241.429007,-0.364,1.5708)/FLY

PATH LIN

MOVE (-70.222008,-120.771996,123.19,-0.528,1.5708)/FLY

PATH CIR

MOVE (0.0,-139.703988,52.959,0.0,1.5708)/
VIA(-49.659007,-130.580003,73.530003,-0.364,1.5708)/FLY

MOVE (70.217995,-120.774003,123.189009,0.528,1.5708)/VIA(49.659007,-130.580003, 73.530993,
0.364,1.5708)/FLY

PATH LIN

MOVE (70.222008,-120.771996,191.77,0.528,1.5708)

WORK 0,00/OFF

SPEED 33.

MOVE (95.355004,-163.995989,191.77,0.528,1.5708)/FLY

MOVE (95.355004,-163.995989,421.769997,0.528,1.5708)

; BURN THE -414 BOSS HOLE AT TVCL...

; A=PI, B=PI/2

SYSDEF (VR[10],VR[11],VR[14],0.0,0.0,0.0)

MOVE (-71.903996,175.548011,400.290005,3.5196,1.5708)/FLY

MOVE (-71.903996,175.548011,242.28999,3.5196,1.5708)/FLY

SPEED 10.

MOVE (-52.961997,129.275002,242.28999,3.5196,1.5708)

WORK 0,00/ON

DELAY .25

MOVE (-52.961997,129.275002,271.179011,3.5196,1.5708)

PATH CIR

MOVE
(-48.362997,131.064991,283.647007,3.4866,1.5708)/VIA(-49.497005,130.640988,282.183002,3.4945,1.570
8)/FLY

MOVE (0.0,139.703988,305.924991,3.1416,1.5708)/VIA(-25.4,137.375011,300.636,3.3266,1.5708)/FLY

MOVE
(48.362006,131.064991,283.647998,2.7956,1.5708)/VIA(25.4,137.375011,300.636,2.9596,1.5708)/FLY

MOVE
(52.96601,129.272995,271.177995,2.7576,1.5708)/VIA(49.497005,130.640988,282.183002,2.7886,1.5708)/
FLY

PATH LIN

MOVE (52.961997,129.275002,213.400996,2.7576,1.5708)/FLY

PATH CIR

MOVE
(48.362997,131.064991,200.932999,2.7916,1.5708)/VIA(49.497005,130.640988,202.397004,2.7886,1.5708
) /FLY

MOVE
(0.0,139.703988,178.654989,3.1416,1.5708)/VIA(25.4,137.375011,183.944006,2.9606,1.5708)/FLY

MOVE
(-48.362006,131.064991,200.932009,3.4876,1.5708)/VIA(-25.4,137.375011,183.944006,3.3226,1.5708)/FLY

MOVE
(-52.96601,129.272995,213.402012,3.5256,1.5708)/VIA(-49.497005,130.640988,202.397004,3.495,1.5708)
/FLY

PATH LIN

MOVE (-52.961997,129.275002,271.179011,3.5196,1.5708)

WORK 0,00/OFF

SPEED 35.

MOVE (-71.917992,175.543007,242.313993,3.5196,1.5708)/FLY

; TRIM OFF THE AFT END...

;

SYSDEF (VR[10],VR[11],VR[12],0.0,0.0,0.0)

;

MOVE (0.0, 178.0, 354.584, 3.1416,1.5708)/FLY

SPEED 10.

MOVE (0.0, 139.7, 354.584, 3.1416,1.5708)

WORK 0,00/ON

PATH CIR

MOVE (0.0, -139.7, 354.584, 0.0, 1.5708)/VIA(139.7, 0.0, 354.584, 1.5708,1.5708)/FLY

MOVE (0.0, 139.7, 354.584, -3.1416, 1.5708)/VIA(-139.7, 0.0, 354.584,-1.5708,1.5708)

WORK 0,00/OFF

SPEED 33.

PATH LIN

MOVE (0.0, 500, 400, -3.1416,1.5708)

SYSDEF

CALL 02

SAMPLE 3

This is a program showing the progression from inch format, to metric, and metric format after processing by RMLCOMP (off-line cutter compensation).

INCH format

Linear values are in inches, angular values are in degrees.

```
; 17 mar 93                ccomp test - portion of customer program

;$DIA=0.020
;$Sright
1          COORD ABSOL
           path ptp
2          SPEED 30
3          MOVE (45, 45, -15, -164.533, -90.09)
4          WORKDEF 2,00 (11)
5          WORK 2,00 /ON
6          WORKDEF 0,0
7          SPEED 15
8          SYSDEF (40.739, 56.190, -22.430, 0.0, 0.0, 0.0)
9          MOVE (0, 0, 40, -164.533, -90.0)
10         path hole (.5)
11         work 0,0/on
12         MOVE (0, 0, 40, -164.533, -90.0)/via(0, -10, 40, -164.533, -90.0)
13         work 0,0/off
14         path lin
15         ;$left
16         MOVE (0, 0, 0, -164.533, -90.0)
```

METRIC format

This shows the conversion of coordinates and angles from inch to metric format. This listing has X, Y, Z, and cutter offset amounts in millimeters. Angular values are in radians.

```
; Metric File C:\CST\CCTEST\17MAR93E.S Created From Inch File  
C:\CST\CCTEST\17MAR93E.SI - Tue Apr 20 13:37:22 1993
```

```
; 17 mar 93                ccomp test - portion of customer program
```

```
;$DIA=0.508
```

```
;$sright
```

```
1          COORD ABSOL  
           path ptp  
2          SPEED 30  
3          MOVE (1143.0, 1143.0, -381.0, -2.871643, -1.572367)  
4          WORKDEF 2,00 (11)  
5          WORK 2,00 /ON  
6          WORKDEF 0,0  
7          SPEED 15  
8          SYSDEF (1034.7706, 1427.226, -569.722, 0.0, 0.0, 0.0)  
9          MOVE (0.0, 0.0, 1016.0, -2.871643, -1.570796)  
10         path hole (12.7)  
11         work 0,0/on  
12         MOVE (0.0, 0.0, 1016.0, -2.871643, -1.570796)/via(0.0, -254.0, 1016.0, -2.87164  
13         work 0,0/off  
14         path lin  
15         ;$cleft
```

Ready For Compilation

The file has been processed by RMLCOMP, and is ready to be compiled for the CRG. Note the value of radius in the PATH HOLE instruction and the formatting of the MOVE instructions where compensation is active, from lines 20-24.

```
; File C:\CST\CCTEST\17MAR93E.CMP Created From File C:\CST\CCTEST\17MAR93E.S - Tue  
Apr 20 13:37:33 1993
```

```
; Metric File C:\CST\CCTEST\17MAR93E.S Created From Inch File  
C:\CST\CCTEST\17MAR93E.SI - Tue Apr 20 13:37:22 1993
```

```
; 17 mar 93                ccomp test - portion of customer program
```

```
;$DIA=0.508
```

```
;$sright
```

```
1          COORD ABSOL  
           path ptp  
2          SPEED 30  
3          MOVE (1143.0, 1143.0, -381.0, -2.871643, -1.572367)  
4          WORKDEF 2,00 (11)  
5          WORK 2,00 /ON  
6          WORKDEF 0,0  
7          SPEED 15  
8          SYSDEF (1034.7706, 1427.226, -569.722, 0.0, 0.0, 0.0)  
9          MOVE (0.0, 0.0, 1016.0, -2.871643, -1.570796)  
10         path hole (12.44600)  
11         work 0,0/on  
12         MOVE (0.0, 0.0, 1016.0, -2.871643, -1.570796)/via(0.0, -254.0, 1016.0, -2.87164  
13         work 0,0/off  
14         path lin  
15         ;$cleft
```

Special Programs

The AMADA "Special programs" are used for various purposes, as documented elsewhere. The listings for programs 1, 2, and 4 follow.

Program 1

- ; 1/15/93 AR's SENSOR CALIBRATE program revised by JDM
- ; the approach to the ref block should be in ABSOL coordinates so that
- ; TCP can be used to comp for change in focus height / tip length.
- ; Program 90 is now TCP - customer can set necessary value from handbox into that program

```
DECLARE X      = VR[43]
```

```
DECLARE Y      = VR[44]
```

```
DECLARE Z      = VR[45]
```

```
DECLARE A      = VR[46]
```

```
DECLARE B      = VR[47]
```

```
DECLARE ZT     = VR[48]
```

```
DECLARE ZMID   = VR[49]
```

```
CALL  2
```

```
LET  X = -105.5           ;5.5           ; -36.480839
```

```
LET  Y = 2217.2          ;1969.694824
```

```
LET  Z = -962.0         ; -960.0         ; -760.5         ; -661.570007
```

```
LET  A = 0.0
```

```
LET  B = 0.0
```

```
LET  ZMID = -600.0      ; -400.0
```

```
LET  ZT = Z + 20.
```

```
sysdef                                     ; cancel any system offsets
```

```
COORD  absol
```

Program 2

This program looks at current machine position, and moves the head to Z-up, A0, B0.

It then turns several program outputs off, cancels any active SYSDEF, and calls program 90 to restore TCP.

Note that this is where the Z-home proximity is active, and typically not at the end of physical travel.

```
;PROGRAM 2 = Z-UP
```

```
DECLARE PRGDATA = OL[22]
```

```
DECLARE PRGNUMB = OL[12]
```

```
DECLARE X      = VR[43]
```

```
DECLARE Y      = VR[44]
```

```
DECLARE Z      = VR[45]
```

```
DECLARE A      = VR[46]
```

```
DECLARE B      = VR[47]
```

```
ASSIGN PRGDATA =0 /8
```

```
ASSIGN PRGNUMB =0 /4
```

```
LET    Z = -50.0      ;50.0
```

```
LET    B = 0.0
```

```
SPEED  40.0
```

```
PATH   PTP
```

```
COORD  ROBOT
```

```
UPDATE IA /ON
```

```
LET    X = IA[0]
```

```
LET    Y = IA[1]
```

```
LET    A = IA[3]
```

Program 4

Program 4 is used to position the machine axes in a convenient "parking position". This is typically in one corner of the machine envelope, above the sensor calibration block with Z-axis at its home position.

```
DECLARE X      = VR[43]
DECLARE Y      = VR[44]
DECLARE Z      = VR[45]
DECLARE A      = VR[46]
DECLARE B      = VR[47]

CALL  2

LET   X = 21.02           ;-36.480839
LET   Y = 2201.2         ;1969.694824
LET   Z = -50.0          ;100.
LET   A = 0.
LET   B = 0.

SPEED  40.0

PATH   PTP

COORD  ROBOT

MOVE   (X,Y,Z,A,B)

CALL  2
```

APPENDICES

Appendix 1

The RML reserved words are listed below. As the language is used on various kinds of robots and machines, not all features are available and/or useful on Optimo.

INSTRUCTIONS

RESERVED WORD	ATTRIBUTES USED	USE
ACCEL		program acceleration rate
ASSIGN		I/O (rarely)
BREAK		similar to handbox function
CALL		gosub a program
CONT		not used on Optimo
COORD		coordinate reference
	ROBOT	robot column - positioning only
	ABSOL	tool tip - contouring
DECLARE		use symbolic names in programs
DEFINE		not used
DELAY		timed dwell
DISABLE		not used
ELSE		logic-flow
ENABLE		not used
ENDTASK		optional
ERR		program error handling
EXTERN		rarely used- linked programs
FUNCTION		not available
GET		not available
GO		jump to next program
GOTO		logic flow within program
GOSUB		logic flow within program
IF		logic flow within program
IMAGE		I/O control - rarely used
INCLUDE		not available
LET		optional
MACHINE		not available
MOVE		machine motion
	TIME	gosub if time exceeded
	START	wait for logic condition
	END	end upon logic condition
	VIA	used with CIR, HOLE
	FLY	maintain velocity across (near) target
	NEXT	not used

RESERVED WORD	ATTRIBUTES USED	USE
	INF	not used
NOBREAK		not used
NOERR		logic flow- cancels ERR - rarely used
NOTRACE		not available
OVERSH		% slowdown at MOVE () / FLY
PATH		type of machine motion
	PTP	coordinated motion- no
	LIN	tool tip follows straight line in space
	CIR	tool tip follows arc in space
	HOLE	round hole on surface
	FRE	not available
PAUSE		program halt
REAL		I/O control- rarely usefull
RETURN		return from GOSUB
SEND		not available
SENSDEF		not available
SENSOR		not available
SETERR		rarely
SIGNAL		not available
SIMUL		similar to handbox function F6
SPEED		program machine speed
STOP		program stop
SYSDEF		program offset / rotate
TASK		not used
TCP		tool tip length compensation
THEN		logic flow (IF / THEN / ELSE)
TIMER	AUTO, DIS	timed gosub
TRACE		not available
UPDATE	ON/OFF	keep groups of I/O current or not
WAIT		rarely
WEAV		no (available-not useful)
WEAVDEF		no (available-not useful)
WEAVING		no (available-not useful)
WORK		laser control
	TIME	not used
	START	wait for logic condition before starting
	END	specify logic condition to halt work
	NEXT	continue program
WORKDEF		define laser parameters for WORK instr.
ZERO		exeute zero axis

ATTRIBUTES

"Attributes" are used with various instructions to modify the action of that instruction. As with instructions, not all attributes are used on Optimo

ATTRIBUTE	Use/ meaning
ABSOL	COORD
AUTO	
C	
CIR	PATH
CLOCK	
DIS	
END	various
FALSE	logic statements
FLY	MOVE
FRE	not used
INF	MOVE
INTR	
LIN	PATH
NEXT	various
NO	equals logic ZERO
OFF	equals logic ZERO
ON	equals logic ONE
HOLE	PATH
PTP	PATH
R	
ROBOT	COORD
S	
START	various
TIME	various
TRUE	equals logic ONE
VIA	MOVE
YES	

MATHEMATICAL FUNCTIONS

These are mathematical functions and logical operators.

FUNCTION	purpose
ABS	absolute value (optional)
ACOS	arc cosine
AND	logical sum
ASIN	arc sin
ATAN	arc tangent
COS	cosine (x)
EXP	e^x
INT	integer value (optional)
LN	natural log
LOG	log (base 10)
MOD	module (x) (optional)
NOT	logical negation
OR	logical sum (inclusive)
PI	constant = 3.14159
SIN	sin (x)
SQRT	square root (x) \sqrt{x}
TAN	tangent (x)
XOR	exclusive - OR

Appendix 2

This appendix includes customer response forms for suggestions, requests, and error / "bug" reports. The Optimo group at Amada welcomes comments and/or suggestions regarding any aspect of the system or documentation.. Feel free to make copies of these forms for actual submission.

SUGGESTIONS

The following item(s) (example: system, manual, etc.)

Should be changed in the following ways: (example: add items to index, revise order, add / revise material, alter machine operation)

Because (examples: can't find information easily, difficult to perform operation, etc.)

Company: _____

Name : _____

Phone number _____

 INDEX

A		FUNCTION	
ACCEL	2-5	from the handbox	9-8
Assignments	7-8, 7-15	instruction	9-1
		passing parameters	9-5
B		G	
Break	6-1	GO	7-1
		from handbox	5-14
C		GOSUB	7-5
CALL	7-2	GOTO	7-4
defining parameters	9-6	H	
from handbox	5-14	Handbox	
passing parameters	9-5	data input	5-2
See also: FUNCTION		editing functions	4-6
comments	1-2	jog motion	4-11
Constants	7-9	HOLE	
COORD		From handbox	5-6
from handbox	5-4	I	
instruction	2-1	IF	7-5
cutter compensation	8-1	Inch/metric	
D		conversion program	8-3
DECLARE	7-1, 8-7	conversion ratios	8-14
Defaults		Instructions	
coord, path, speed	5-4	adding	6-8
DEFINE		deleting	6-7
syntax	9-6	locating	6-7
DELAY	3-1, 3-5	selecting	6-7
Dynamic	2-5	J	
E		Jog motion	4-11
ENDTASK	3-5, 9-1	L	
ERR	3-5	Laser	
Expressions	7-12	differences	5-10
F		power control	3-2 - 3-3
Feedrate		Laser schedules	3-1
during FUNCTION	9-8	LET	7-15
selectable	4-4	from handbox	4-8
Fly		Listfile	6-7
from handbox	5-8		
usage	5-8		

M

MOVE

from handbox 5-7
 instruction 2-6
 modifiers 2-6, 5-8

O

Outputs

assignments 7-10, 7-17
 from handbox 5-13

OVERSH

2-5
 from handbox 5-9

P

Part-program

deletion 4-13
 format 1-1
 length 1-1, 9-1
 organization 5-2
 selection 4-12
 testing 6-5

PATH

from handbox 5-4
 instruction 2-2

PAUSE

3-5

Program control

call, go 7-1
 goto, gosub 7-4
 if, then, else 7-5
 sequence numbers 1-2

Program testing

break 6-1
 function keys 9-7
 simulation 6-2

Programming

from handbox 5-1

R

RETURN

7-5

S

Sensor

usage 5-3

SETERR

3-6

Shutter

control differences 5-10

Simulation

6-2

Speed

2-5

approaching workpiece 5-3

from handbox 4-4, 5-7

Start key

GO, CALL 5-14

STOP

3-5

Symbols

declare 7-1

SYSDEF

7-6

Teach mode 5-1

T

TCP

from handbox 4-14

U

units

determining 8-5

V

Variables

7-10, 7-16

linear/rotary 8-6

W

WORK

3-2

from handbox 5-10

WORKDEF

controlling power 3-1, 5-12

defined 3-1

Z

Zero Axis

from handbox 4-10