REAL-TIME CONTROL SYSTEM MODIFICATIONS FOR A DEBURRING ROBOT USER REFERENCE MANUAL

by

Karl N. Murphy

August 4, 1988

I. INTRODUCTION

At the National Bureau of Standards' Automated Manufacturing Research Facility (AMRF) a PUMA 760 robot deburrs metal parts at the Cleaning and Deburring Workstation (CDWS). The robot is controlled by the NBS developed Real-Time Control System (RCS). The basic RCS, as described in [1], was extended to meet the needs of the workstation and this manual explains these additions.

This section of the manual gives a brief description of the workstation's operation, explains how the manual is organized, describes who should use this manual, and enumerates the documentation convention used.

1. SYSTEM DESCRIPTION

The CDWS cleans and deburrs the metal workpieces that were machined at the AMRF. Equipment at the workstation includes a PUMA 760 robot for deburring, a Unimate 2000 robot for part handling and buffing, a washer/dryer system for cleaning, a buffing wheel system for buffing, a rotary vise for clamping parts, and two roller tray stations for receiving parts. The PUMA 760, a six-axis electric robot, is fitted with a quick change wrist which allows for selection between a chamfering tool, an end brush, and a hole brush. The workstation floor plan is shown in Appendix A.

One of the principal advances of the workstation is a system that generates accurate deburring paths from CAD data [2]. The process is shown in Figure 1. A graphic representation of a part is generated from geometry data and presented to a user who selects the edges to be deburred and the desired deburring parameters. A robot path planner generates an initial deburring path which is sent to the robot.

Unfortunately, this initial path is not suitable for deburring due to several sources of inaccuracy. The inaccuracy of the robot contributes most significantly. Although most robots have high *repeatability*, the ability to return to the same location when given the same coordinates, the precision with which a computed point can be attained, the *accuracy*, is low. The PUMA 760 robot's repeatability is 0.005 inch, while errors in accuracy of 0.25 inch may occur for motions about the part. Errors in machining, large part tolerances, and tool wear also contribute to discrepancies between the software models and the real workstation. To compensate for inaccuracy, the robot corrects the computed path by approaching each endpoint half an inch back along the tool axis, driving its tool forward while reading a wrist force sensor until a desired contact force is met. The coordinates of this point replace the computed coordinates. When the entire set of endpoints on the path has been corrected for robot inaccuracy, the paths are traced with the selected tool to accomplish deburring.

The control structure for the workstation is hierarchical, as shown in Figure 2. At the top is the workstation controller (WSC) [3, 4]. The WSC accepts commands from either an operator at the workstation or from a remote source, such as the AMRF cell controller. The WSC sends commands and data to the two robots and to the washer/dryer system. The PUMA 760 robot is controlled by RCS [1] which also controls the deburring tools and the quick change wrist. The Unimate 2000 robot is controlled by VAL, the vendor supplied controller [5, 6]. The VAL controller sends commands to the tray stations and to the buffing wheels. Both the PUMA 760 RCS and the Unimate 2000 VAL controller can send commands to the rotary vise but only one system has control of the vise at a given time. Arbitration is coordinated by the WSC. The WSC does not control the vise directly because the WSC control delay is too large.

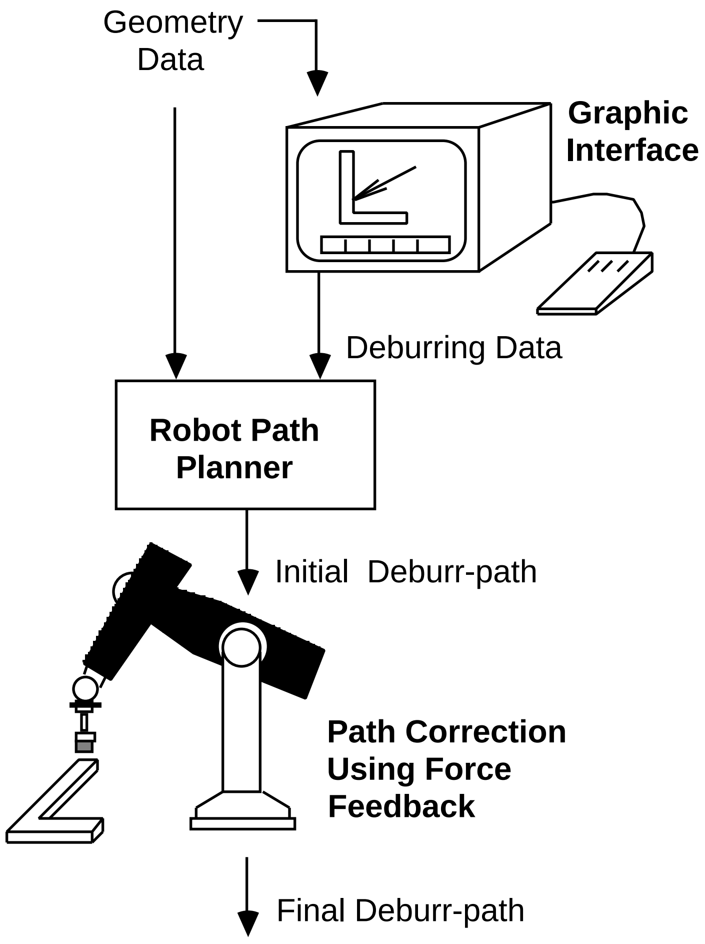


Figure 1. CAD Directed Robotic Deburring.

The RCS as configured for the PUMA 760 robot is shown in Figure 3. The four control levels, Task, Path, and Primitive (Prim), form a control hierarchy with Task level at the top and Prim level at the bottom. Each control level decomposes commands from the level above and generates commands for the level below. The output of Prim is a sequence of joint angles which are sent to the joint servo cards over a serial link every 28 milliseconds. The data required by the control levels (robot paths, trajectory parameters, tool descriptions, locations, joint limits) is stored in the RSL database in common memory.

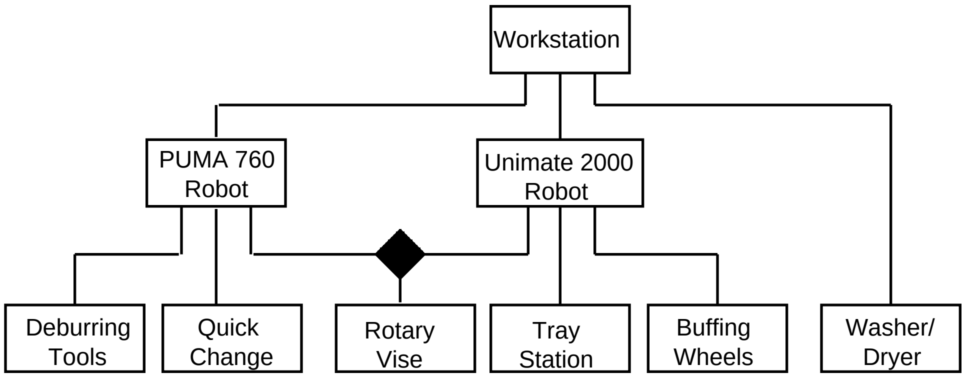


Figure 2. CDWS Control Hierarchy

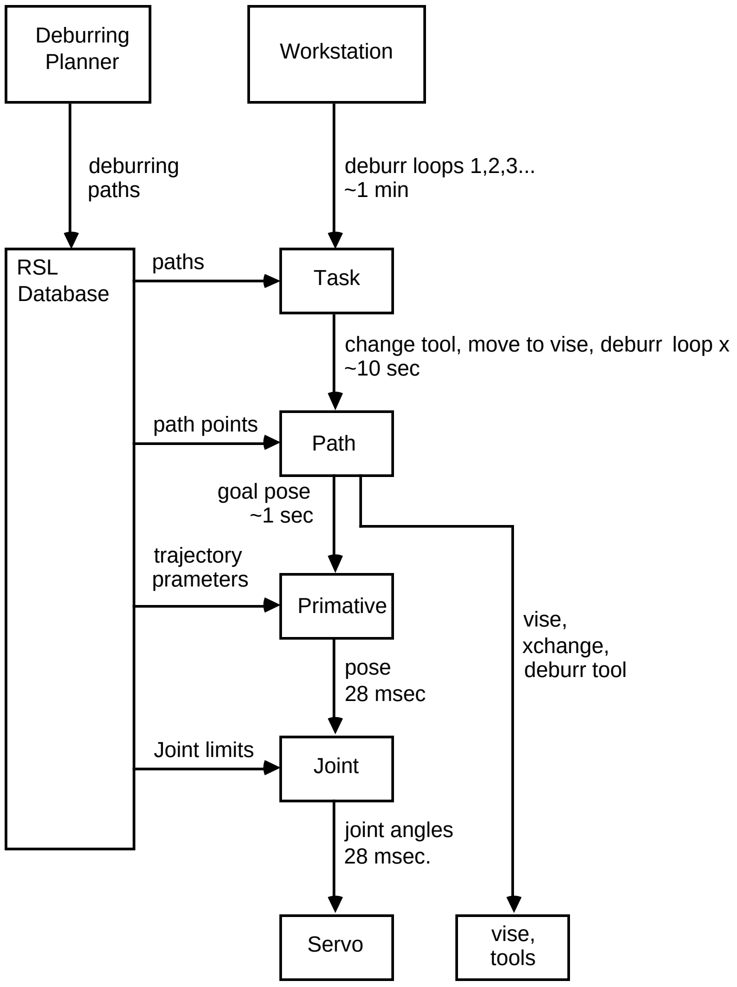


Figure 3. The Real-time Control System (RCS) for the PUMA 760.

2. HOW THIS MANUAL IS ORGANIZED

This manual explains the changes made to the basic RCS at the CDWS. The reader should be familiar with the basic RCS, described in [1], before reading this document.

Chapter II, Basic Operations, explains basic operations that the operator commonly uses while running and modifying the RCS. These include entering commands to the system, determining system status, entering data, and defining locations.

Chapter III, RSL Changes, explains the modifications made to the RSL board. These include both separating the RSL board from the master board so that it can run in the background mode and the addition of new path types, new path point types, and a new tool defining word.

Chapters IV thru VI, Task Changes, Path Changes, and Prim Changes, describe the modifications made to the three top control levels. No changes were made on the Joint Level. These changes include an interface to the workstation controller and the addition of new commands and routines necessary for deburring.

Chapter VII, Communication Changes, describes the modifications to the communication board. This board handles the communication between RCS control levels and between the Joint level and the PUMA 760 Robot. Certain timing changes were made to include a wrist force sensor and to reduce delay.

Chapter VIII, Workstation Interface, details the interface between Task level and the workstation controller (the SUN) and between RSL and the robot path planner (also the SUN). The low level communication protocol is given.

Appendix A lists the hardware configuration for the robot controller and Appendix B gives safety and power up instructions.

3. WHO SHOULD USE THIS MANUAL

This manual is intended for NBS personnel and for government, industry, and university researchers who need to understand, operate, and modify the RCS for the CDWS' PUMA 760 Robot. This manual should be viewed as an addendum to "The NBS Real-Time Control System / User Reference Manual" [1] which provides a detailed description of the basic RCS. The reader should have a through understanding of RCS, the Smacro Programming language, and the robot control application RSL before reading this manual.

To power up and demonstrate the entire workstation, see [7]. Other documents explain the workstation controller [3, 4], the control of the Unimate 2000 robot [5, 6], and CAD Directed Robotic Deburring [2]

The Cleaning and Deburring Workstation is an ongoing research effort. Several aspects of operation will change without notice. If the operator encounters an error that can not be resolved, he should contact one of the principle operators for assistance.

4. DOCUMENTATION CONVENTION

This manual uses the following conventions:

- `The name of each keyboard key mentioned in the text appears in uppercase courier letters.  For example, the carriage-`

return key appears as RETURN.

- In a control-key sequence, a caret (^) represents the CONTROL key. For example, control-C appears as ^C. Hold the CONTROL key while you press C.
- Information that appears on the terminal screen appears in bold courier print. For example, the **:R rsl>** prompt appears on the 760 RCS terminal when the RCS system is awaiting input from the keyboard.
- Information that you must enter exactly as it appears in the manual is in plain courier print. For example, "Enter 3 LOAD" means to type 3 LOAD and press RETURN.
- Variable data that you must enter is represented within square brackets ([ ]). The instruction "Enter [block #] LOAD" indicates that you are to replace [block #] with a specific block number when you enter the load command. For example, you might type 3 LOAD and press RETURN.
- Optional data that you may omit is represented within curly brackets ({ }). Most optional data is also variable data and appears in both curly and square brackets. The instruction "Enter RECEIVE_TRAY { ([Tray_serial_#]) }" indicates that you may either omit the serial # or specify a desired serial #. For example, you might type RECEIVE_TRAY and press RETURN or you might type RECEIVE_TRAY (123) and press RETURN.
- When input is to be typed in a specific window or on a specific board, the actual prompt is included in the command string. For example, on the 760 RCS system the instruction "Enter **:R Mst>** Halt" means to type Halt and press RETURN on the master board.

II. BASIC OPERATIONS

This chapter describes certain commands that the operator uses while running the CDWS RCS and while making minor changes. These include general commands, monitoring the system, entering data, and new smacro words. To power up the robot and RCS, see the power up instructions in Appendix B.

1. GENERAL INFORMATION

DIFFERENT BOARDS: The RCS system has 6 boards. There is a master board, **`Mst>`** , and 5 slave boards, **rsl `>`**, **`t&p>`** (Task and Path), **`prim>`**, **`joint>`** and **`comm>`**. You must enter different things on different boards. In these directions, the prompt tells you the board to enter commands on. For example: **`Mst>`** `GO` means type a `G`, an `O`, and `RETURN` on the **`Mst>`** board. On the computer screen, the prompt shows the name of the board you are on. Some commands to the master board send commands to slave boards, ex: **`Mst>`** `GO`.

To move from board to board use the function keys. You can move back and forth between the master board and any slave board. To move between two slave boards, you must move to the master board first. The RCS terminal has 11 function keys labeled `f1` thru `f11`. Each has a separate function as labeled on the duct tape. For example, `f3` will move you to the **`Mst>`** board.

JOYSTICK: To enable the joystick, enter either **`Mst>`** `JOYSTICK` or **`t&p>`** `912 LOAD`. These commands clear certain errors and enable the joystick. To move the robot, flip the enable switch and set the velocity to a low value. The most common motions are tool X, tool Y and tool Z. See Figure 4.

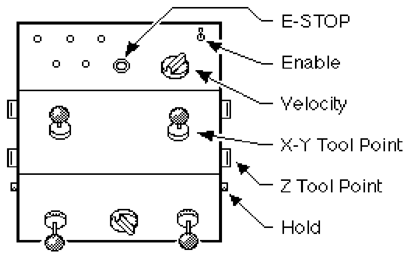


Figure 4. Commonly used RCS joystick switches

STATUS: Enter **`t&p>`** `STAT`. The status of the robot will be printed.

COMMAND DIRECTORIES: **`t&p>`** `910 LIST` and **`t&p>`** `920 LIST`. These blocks list different commands to the robot. Ex: **`t&p>`** `924 LOAD` moves to VISE-SAFE. If you would rather enter the commands from the terminal, use **`t&p>`** `EXEC [command string]`. This copies the command string into Task's input buffer.

SUN COMMANDS: **`t&p>`** `ENABLE-SUN-CMD` and **`t&p>`** `DISABLE-SUN-CMD` tells Task level to accept/ignore commands from the workstation controller.

VISE: **t&p>** `GET-VISE` locks the 2000 out of the vise volume. **t&p>** `DROP-VISE` drops the vise volume request. The 2000 can now get the vise.

RSL: At power up, the RSL board is told to accept robot paths from the SUN. This program is called: **rsl>** `BGO`. If RSL is to ignore the paths from the SUN enter: **Mst>** `ABT_RSL` **rsl>** `DGO`. You will not get a prompt back. To abort these processes, `BGO` or `DGO`, enter: **Mst>** `ABT RSL`.

2. MONITORING THE SYSTEM

There are two commands used to monitor the system.

Rsl> `HELP` lists several commands used to examine RSL files.

t&p> `STAT` prints the status of the system, including parameters such as current location, current command, current tool, SUN command status, vise owner, current error, etc.

3. ENTERING DATA INTO THE SYSTEM

To define poses, two commands are used. These are REC-POSE and CALC-POSE

REC-POSE [pose-name] is the standard RSL version. Just move the robot to the desired pose with the desired tool, and give the command to RSL. If the pose name already exists, it is redefined to the new pose.

`CALC-POSE [pose-name]` is used to define poses that require accurate rotational alignment. Put the robot into joystick mode and give the command to RSL. RSL will tell you to move the robot first to the origin of the pose, then to the Y axis of the pose, and then to the X axis of the pose. The orientation is then calculated from the X and Y axis. If the pose name already exists, it is redefined to the new pose.

Some tools, such as the countersink tool, require accurate measurement of the tool offset. Measure the offset with a dial gage by noting the deflection when the tool is rotated 180° about the tool axis, normally the Z axis. See Figure 5.

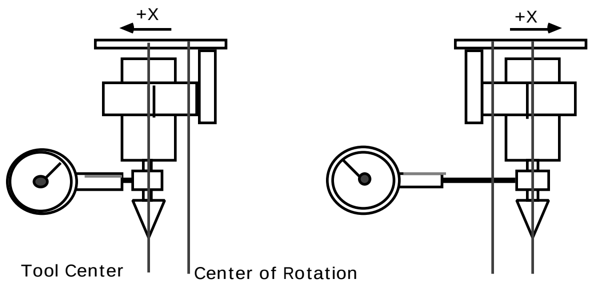


Figure 5. Measuring tool offset. The offset is half of the gage's deflection.

4. SMACRO EXTENSIONS

A few new smacro words are defined:

`MSEC (# -- )`	is a FORTH word that delays the specified number of msec. (±20%).
PMSEC (# --)	is the same as MSEC only PAUSE is called each cycle of the delay loop. The timing is accurate when one background loop is active.
`~MSEC (# -- )`	is a smacro version of MSEC.
~PMSEC (# --)	is a smacro version of PMSEC.
cr	is a smacro word that prints a carriage return. (It takes less space than `~F CR`.)

III. RSL CHANGES

This chapter describes the changes made to the RSL Board on the PUMA 760 RCS System in the Cleaning and Deburring Workstation. These include 1) system changes, 2) new defining words for paths, path points, loc types, tools, and 3) an explanation of the file structure.

1. SYSTEM CHANGES

The major changes to the RSL board are that it is separated from the master board and that `BGO`, a program that accepts deburring paths from the offline path planner residing on the SUN 3/160, is added. The SUN connects to the RSL board by a serial link. See Chapter VIII for details.

By removing the RSL code from the master board and running it on a new board, the user can monitor the system and perform other master board functions while RSL is accepting data from the SUN. Also, the master board now has a small amount of user code so the D>M's on the master board do not change very often. Thus the system can be loaded (after base load) by just typing `1480 0LOAD`. Also, all D>Ms on the master are good for any `PRESERVE` / `RESTORE`.

On the Master board, the `D>M`'s 1 thru 4 are the same memory images but the auto load blocks are different. Auto load block 1 does nothing. Auto load block 2 does a 1 `RESTORE`. Auto load block 3 does a 9 `RESTORE`, `BOOT-SYSTEM`. Auto load block 4 does a 9 `RESTORE`, `BOOT-SYSTEM`, `Rsl> BGO` and `Mst> POWERUP`. The `5 D>M` is the standard memory image with editor, tape, and print functions. When you just want to run the system, enter `4 D>M`. If you want to boot the system without powering up the robot enter `3 D>M`. `1 D>M` is used to restore just the master board after a `5 D>M`.

On the RSL board, `1 D>M` is base, `2 D>M` is RSL, and `3 D>M` is RSL running `BGO`.

`BGO` is a routine on the RSL board that accepts deburr paths from the SUN. More precisely, it executes the ASCII strings sent by the SUN by copying the strings into ram and calling RAM-LOAD. This allows `BGO` to use the RSL defining words directly without modifying them or developing a second copy for use by the SUN. The SUN sends down the RSL code that a user normally types into a block. In fact, the SUN can send down any command that you would type or load from a block, such as `3 D>M`. Normally, the SUN only sends down deburr paths.

`BGO` on RSL is not a background task unlike on other boards. This is due to several reasons. Smacro files cannot be accessed by both the foreground task and a background task, so a user could not use RSL when `BGO` is running. RSL defining words normally print messages to the screen; something background tasks cannot do. And RSL defining words call `abort` which would kill the background task.

To handle errors, `BGO` revectors `QUIT` so that after clearing the error, `QUIT` calls `BGO`. This way if bad code is sent down from the SUN, `ABORT`, which calls `QUIT`, jumps back into `BGO`. To stop `BGO`, enter **`Mst>`** `ABT RSL`. This also vectors `QUIT` back to its original definition. Typing `^C` directly to the RSL board (using the terminal switch box) has the same effect.

Two other commands have related functions. `DGO` (Dummy GO) carries out the handshake with the SUN but does not load the command strings. This is useful if you want the RCS system to ignore the new paths being sent from SUN. `STORE-RSL ( block # -- )` stores the strings sent from the SUN, one string per block starting at block (13000 + `block`#). **`Mst>`**

`ABT RSL` aborts these commands.

2. NEW PATHS

Extensions to RSL on the CDWS 760 RCS will allow deburring paths and quick change paths to be defined. The two -path- types are called deburr-loop and qc. Several new -ppt- 's are defined in Section 3.

DEBURR-LOOP PATH

The deburr-loop path has the following syntax:

```
-path- deburr-loop ["part-name"] [loop#] ["vise-loc"] [part-pose]
      [traj-type] [traj-prams] ["tool"] [tool-force] [tool-speed]

-ppt-  acquire-vise
-ppt-  goto goal# [approach-pose] [traj-type] [traj-prams]
-ppt-  tool-on
-ppt-  vertex   [vertex-pose]
-ppt-  vertex   [vertex-pose]
      .
      .
      .
-ppt-  vertex   [vertex-pose]
-ppt-  goto goal# [approach-pose] [traj-type] [traj-prams]
-ppt-  tool-off
```

where:

`"part-name"`	is the name of the part. Type: string.
`loop#`	is the loop number. Type: integer.
`"vise-loc"`	is the name of the location of the vise, such as: `VISE0`, `VISE90`, and `VISE180`. These three will be previously defined. Type: string.
`part-pose`	is the pose (quaternion, translation) of the part in the vise frame. Type: 7 floating point numbers. Units: translation in inches
`traj-type`	is the trajectory type. For now use the string `cart` for cartesian. Type: string
`traj-prams`	are the trajectory parameters as defined in RSL. For a `cart` trajectory type (see above), there are six parameters: `acc-max` (in/cyc/cyc), `vel-max` (in/cyc), `neighborhood` (in), `rot-acc-max` (deg/cyc/cyc), `rot-vel-max` (deg/cyc), `rot-neighborhood` (deg). A cycle is 28 msec. Typical values are: `0.03 0.30 1.00 0.25 1.50 6.00`. Type: 6 floating point numbers.
`"tool"`	is the name of the tool. These are currently three tools: `T1-EB120`, `HB-5/8`, and `CS`. Type: string.

`tool-force`	is the magnitude of force that the tool will push against the part when applicable. Type: floating point. Units: oz.
`tool-speed`	is the rotational speed of the tool. During deburring, air valves will be opened so that the free speed will be the closest speed obtainable for the given valve/restrictor setup. However, the SUN should have a list of obtainable speeds for each tool. Type: floating point. Units: RPM
`vertex-pose`	is the pose of an individual vertex in part coordinates. Type: 7 floating point. Units: translation in inches.
`approach-pose`	is the approach and depart pose (quaternion, translation) in part coordinates. There will normally be at least one goto -ppt- at the beginning and end of each LOOP. Additional goto -ppt-'s can be used at the beginning, at the end, or in the middle of the LOOP. Type: 7 floating point. Units: translation in inches.

Poses specify both rotation, a quaternion, and translation. Poses have the form `[c/2 xs/2 ys/2 zs/2  x y z]` where `c/2` is the cos of the half angle, `xs/2 ys/2 zs/2` is the sine of the half angle times the unit rotation vector, and `x y z` is the translation. Type: 7 floating point. Units: translation in inches.

The relation between the vise location, part location, vertex frame, and goal# frame are shown in Figure 6.

An example deburr path is:

```
-path-  deburr-loop bracket-19 4 VISE-90 1. 0. 0. 0. 1. 0. 2.
    0.03 0.3 1.0   0.25 1.5 6.0 Brush-2  10. 20000.

-ppt-  acquire-vise
-ppt-  goto  goal# 1. 0. 0. 0.   0. 0. 4.
    cart 0.03 1.3 1.0   0.25 1.5 6.0
-ppt-  tool-on
-ppt-  vertex   1. 0. 0. 0.   0. 0. 2.
-ppt-  vertex   1. 0. 0. 0.   1. 0. 2.
-ppt-  vertex   1. 0. 0. 0.   1. 4. 2.
-ppt-  vertex   1. 0. 0. 0.   0. 4. 2.
-ppt-  vertex   1. 0. 0. 0.   0. 0. 2.
-ppt-  goto  goal# 1. 0. 0. 0.   0. 0. 4.
    0.03 1.3 1.0   0.25 1.5 6.0
-ppt-  tool-off
```

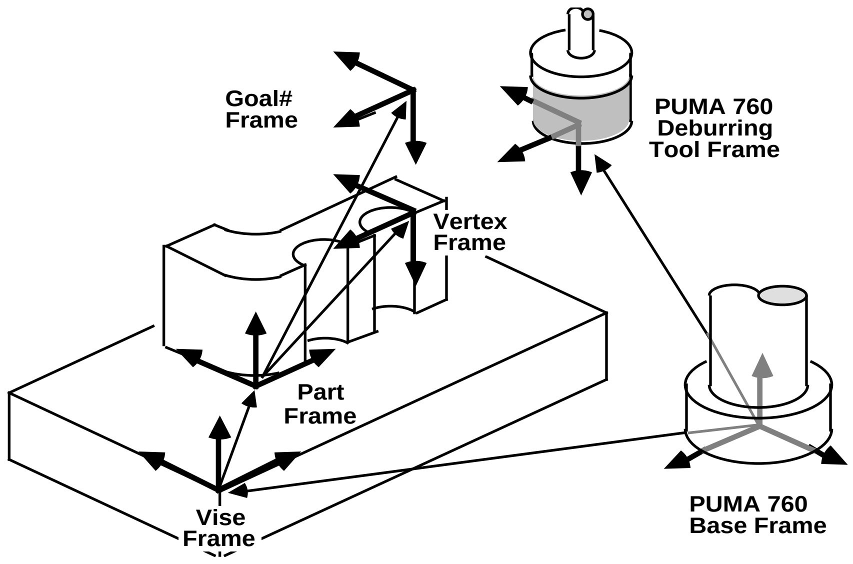


Figure 6. The various frames used for deburring.

To redefine a loop with new values, send a path with the same part name and loop number to RSL. To delete an individual loop or all the loops on a part, send one of the following to RSL:

```
    DELETE-LOOP   ["part-name"]  [loop#]

    DELETE-PART   ["part-name"]
```

While Task level is executing, RSL can define new loops, redefine old loops, or remove old loops as long as the loops are not referenced by the commands sent to Task.

QC PATH

There is one quick change path, `qc`, that RCS uses to change tools. It is the same path regardless of the tools being exchanged. The path parameters, such as `old-tool-^` and `new-tool-^` are set by Task at run time. The quick change path has a simple syntax:

```
-path- qc
-ppt-  [ path point ]

      ...

-ppt-  [ path point ]
```

3. NEW PATH POINTS

Several new path points are defined. The path points are complied into RSL files on the RSL board. The resulting actions of the various control levels at run time are mentioned.

`tool-on` and `tool-off`, turn the tool on and off. The valve setting for turning the tool on is stored in the path header.

acquire-vise requests the vise volume and waits for the acknowledge. The robot will stop moving. No errors are reported.

release-vise drops the vise volume request.

qc-lock locks the quick change wrist. The robot delays to allow the valves to actuate.

qc-unlock unlocks the quick change wrist. The robot delays to allow the valves to actuate.

set-tool ["tool-name"] sets the current tool to "tool-name". "tool-name" must be the name of a tool as defined by -tool-. This tell the system what tool is on the wrist, and what tool offset to use. This -ppt- is normally used in quick changes.

qc-slot? [old/new] [full/empty] checks the proximity switch in the quick change rack for either the old or new tool slot. This will report an error if the slot is not full/empty.

qc-id? [old/new/null] checks the identification code on the quick change tool currently on the wrist. This will report an error if the id is not the correct id for the old/new tool

vertex [vertex-pose] compiles the pose vertex-pose. The pose is in part coordinates and is normally on a part edge. At run time, Path level does different things depending on the current tool and command from Task (teach or deburr).

goal# is a new location type used in deburr-loop paths. The syntax is:
goto goal# [approach-pose] [traj-type] [traj-prams]. approach-pose is the pose of an individual vertex in part coordinates. The trajectory types and parameters are the standard RSL types. This loc type is used to move the deburring tool to safe approach and depart locations before and after deburring. The trajectory parameters allow the planner to move the robot around the part at a different speed from the feed speed specified in the header of the deburring path.

Both vertex and goal# call DBR-ADJUST-POSE? . If the flag dbr-adjust-pose? is true, then DBR-ADJUST-POSE? rotates the quaternion about the Z axis of the tool so that the X axis of the tool is as close to the vector x-des-base as possible. This kludge prevents the tool from twisting up. If a tool should not be rotated about the Z axis, set dbr-adjust-pose? to false.

4. TOOL DEFINING WORD

Tools are defined using the syntax:

```
      -tool-    ["name"] [wrist>tool-pose] [qc-id#] [rack-loc-
name] [slot] [angle] [valve-set & speed ( 7 pairs )] [weight]
[interference vector]
```

where:

["name"]	is the name of the tool. Type: string.
[wrist>tool-pose]	is the pose from the wrist to the tool. Type: 7 floating point numbers. Units: translation in inches.

[qc-id#]	is the identification number of the tool. Type: integer.
[rack-loc-name]	is the name of the location of the quick change slot. Type: string.
[slot]	is the quick change slot number, 1, 2 etc. Type: integer.
[angle]	is the angle about the Z axis of the tool in the quick change rack. This rotation is added after all mtb's have been added. This way tools with different angles can all use the same quick change path. Type: Floating point number. Units: Degrees
[valve-set & speed (7 pairs)]	is 7 pairs of a valve setting and a rotational speed. Type: 7 pairs of integer, floating point. Units: Speed in RPM.
[weight]	is the weight of the tool. Type: floating point. Units: oz.
[interference vector]	is vector that is added to a vertex when the robot is teaching itself. Type: 3 Floating point. Units: inches

5. RCS INTERNAL FILE STRUCTURE

The new RSL files required are:

DEBURR-LOOP-FILE (set by the path "deburr-loop")

```
VAR-O deburr-loop-var
   15  strv    "dbl-part-name"      % part name
       iv      dbl-loop#            % loop number
       iv      dbl-vise-loc-^       % pointer to location file

    7  1f:a    dbl-part-pose        & location of part in vise coordinates
       iv      dbl-tool-^           % pointer to tool file
       iv      dbl-traj-type        % trajectory type, assumed to be cart

       iv      dbl-traj-^           % pointer to traj-file
       fv      dbl-tool-force       % cutting force
       iv      dbl-valve-set        % setting to turn tool on

       iv      dbl-path-L^          % pointer to path file list


TOOL-FILE
   15   strv   tl-"tool-name"       % string name
    7  1f:a    tl-wr>tl-pose        % pose adjusted as brushes wear
    7  1f:a    tl-tl>wr-pose        % pose calculated from above pose

       iv      tl-qc-id             % 1,2,3,etc
       iv      tl-loc-^             % pointer to location of rack slot
    7  1:a     tl-valve             % valve setting ( 1 - 7 )
    7  1:fa     tl-speed                    % free speed of tool at above setting
```

```
VERTEX-FILE   % vertex's   ( set by the path "deburr-loop" )
    7  1f:a     vertex                      % pose of vertex.
```

New entries in current files are:

```
LOC-FILE
        add loc's:    VISE-0, VISE-90, VISE-180, QC1, QC2, QC3, QC4,
QC5.
POSE-FILE
        add pose:     VISE-0, VISE-90, VISE-180, QC1, QC2, QC3, QC4,
QC5.
CART-TRAJ-FILE
        add traj:     traj values as defined in -path-
```

IV. TASK CHANGES

This chapter describes the changes to the RCS Task level on the PUMA 760 at the Cleaning and Deburring Workstation. These include a command and status link to the workstation controller and new Task commands, DEBURR-LOOP, TEACH-LOOP, QC, and JOYSTICK.

1. WORKSTATION CONTROLLER INTERFACE

There are two RS-232 lines between the workstation controller (the SUN) and the Task level, one command and one status. See Chapter VIII for protocol and hardware specifics.

There are two routines which stuff the Task input command buffer, SUN-CMD and MON-CMD.

The interrupt driven routine SUN-RX receives command strings from the SUN. When a complete command has been received, SUN-RX sets a flag. SUN-CMD then copies the string into the Task command buffer, increments the command number, and clears the flag. The routines ENABLE-SUN-CMD and DISABLE-SUN-CMD enable/ disable the interrupts.

Commands are entered from the terminal using `EXEC [command-string]`. This routine sets a flag that MON-CMD waits for. When the flag is set, MON-CMD copies the string into the Task command buffer, increments the command number, and clears the flag.

2. NEW COMMANDS

There are several new commands that Task accepts: DEBURR-LOOP, TEACH-LOOP, QC, and JOYSTICK.

<u>DEBURR-LOOP and TEACH-LOOP:</u>

To the Task level, these commands are almost identical. The only difference is that Task sets the flag `teach-vertex?` either to true or to false. This flag is part of the command to Path level and tells Path to execute in either the deburr mode or in the teach mode. See Chapter V.

To command the robot to deburr one or more loops, issue the command:

```
DEBURR-LOOP ["part-name"] [loop#] {[loop#]} ... {[loop#]} ;;
```

To run the self-correction routine, issue the command:

```
TEACH-LOOP ["part-name"] [loop#] {[loop#]} ... {[loop#]} ;;
```

The double semicolons are needed.

During a DEBURR-LOOP command or a TEACH-LOOP command, Task cycles through the list of loops. For each loop, Task determines whether to change tools, when to deburr, and when to move between the vise, the quick change rack, and the vise safe location. To do this, Task uses the following state table

State			Action
current location	current tool	last loop ?	
not at tool rack	not deburr tool	no	Move to tool rack.
at tool rack	not deburr tool	no	Change tool.
not at vise	deburr tool	no	Move to vise.
at vise	deburr tool	no	Deburr loop.
xx	xx	yes	Move to safe

xx = don't care

QC (Quick Change)

The QC command tells the robot to change tools. Task first moves the robot to the qc rack. If the current tool is incorrect, then Task commands the QC path.

JOYSTICK

This command performs like the old PAUSE command. It sends the command JOYSTICK to Path level and clears the current location pointer (sets it to null). On the Path level, PAUSE no longer activates the joystick. The commands PAUSE and JOYSTICK are separated so that the SUN can send a pause command during error conditions without activating the joystick.

V. PATH CHANGES

This chapter describes the changes to the RCS Path level on the PUMA 760 at the Cleaning and Deburring Workstation. The changes include the addition of several new path points and modifications to the control level.

1. PATH POINTS

`vertex`

The vertex path point is the main path point used by the deburring path. A vertex path point is stored as a pose in part coordinates. See Figure 6 in Chapter III. The Path level does several different functions depending on the deburring tool and if the robot is in the teach mode or the deburr mode.

In the teach mode, only the endbrush is taught. All other tools behave the same in the teach mode as in the deburr mode. Teaching with the end brush does the following. The robot moves to the vertex with a half inch offset along the Z axis. The robot delays a certain number of cycles to allow vibrations to die out. The force sensor is nulled. The robot moves in along the Z axis until the restrictive force is equal to the deburring force specified in the deburr loop path header. The controller calculates the current location of the tool with an interference offset in part coordinates and substitutes this location for the original vertex. The robot then moves back to the start point. If the desired restrictive force is not detected after an inch of travel, the robot stops and reports a part missing error.

In the deburr mode, all tools except the counter sink tool simply move to the vertex pose. The trajectory parameters are specified in the deburr loop path header. If the tool is a counter sink tool, the robot pauses at the last commanded position, normally specified by `goal#` as a position just above the vertex point. The force sensor then nulls itself. The robot moves in until either the vertex is reached, or the restrictive force is equal to the deburring force as specified in the deburr loop path header.

All the decisions on what each tool does during the teach mode and deburr mode are hard coded in the vertex routine. In a more desirable operation, the various procedures would be specified in a separate vertex path. Each tool would have two such paths for both the teach and deburr mode. The run time input to these paths would be the pose specified in the vertex path point. Unfortunately, the current control structure does not allow a path point to call a new path.

`goto goal#`

`goal#` is actually a location type. The robot moves to the `goal# pose` in part coordinates. See Figure 6 in Chapter III. The specified trajectory parameters are used. This is used to move the robot to safe approach and depart locations during deburring.

`tool-on` and `tool-off`

These path points turn the tool on and off. The on valve position is stored in the deburr loop path header. For operation from the terminal, use `[valve set (1-7)] T-ON` and `T-OFF`. See Chapter II.

qc-lock and qc-unlock

These path points lock and unlock the quick change wrist. The robot delays to allow the valves to actuate. For operation from the terminal, use X-LOCK and X-UNLOCK. See Chapter II.

qc-slot?

This checks the proximity switch in the quick change slot of either the old or new tool to see if the slot is full/empty. The path point is done when the slot is full/empty. This accounts for any bouncing of the switch. After a certain delay, an error is reported.

qc-id?

This path point reads the identification code on the quick change wrist check for the proper tool on the wrist or for no tool. The path point is done when the correct tool is on/off the wrist. After a certain delay, an error is reported.

set-tool

This sets the current tool pointer to the commanded tool. This is only a change in the state of the controller. No hardware changes take place. This tells the controller what tool offset to use, etc.

acquire-vise

This sets the vise volume request and waits for acknowledgment. The robot will stop at the last commanded position. No errors are reported. For operation from the terminal, use GET-VISE. See Chapter II.

drop-vise

This drops the vise volume request. For operation from the terminal, use DROP-VISE. See Chapter II.

2. MISCELLANEOUS

Force Limits

The PRE-PROCESS level now monitors the force readings. The maximum forces and torques are stored in the variable force-max. If the force in any direction exceeds the corresponding limit in the array force-limit, then the robot backs away from the force, stops, and reports an error.

Global Speed

To change the speed of the robot for all paths, use the SPEED command. This sets the variable speed. In post-process, all trajectory parameters are scaled before sending them to Prim. The SPEED command stores the new value in Path's initialization blocks. This way a speed change stays in effect even after turning the controller off and on.

VI. PRIM CHANGES

This chapter describes the changes to the RCS Prim level on the PUMA 760 at the Cleaning and Deburring Workstation.

Prim accepts a tool pointer from Path level every cycle. This pointer specifies the tool offset that Prim uses for all motions. Prim keeps track of the wrist position, velocity and acceleration and transfers them to the tool every cycle. Thus the tool pointer can be changed while the robot is moving.

Prim can change the wrist configuration during cartesian motions. This produces a smooth straight line motion without wrist flips at configuration boundaries.

VII. COMMUNICATION LEVEL CHANGES

This chapter describes the changes to the RCS Communication level on the PUMA 760 at the Cleaning and Deburring Workstation. This includes changes to the timing and the addition of code to read the wrist mounted force sensor.

1. Timing

Three functions occur during each control cycle on the communications board: (1) the force sensor is read and the data is put in common memory; (2) the RCS interboard communication takes place; and (3) the slave communication runs. The least amount of control delay occurs when these functions are sequenced in that order. The force data in common memory is updated just before the beginning of a control cycle, denoted by the RCS communications cycle. Also, the commanded joint angles arrive from the joint level during the communication cycle slightly before they are sent to VAL. See Figure 7.

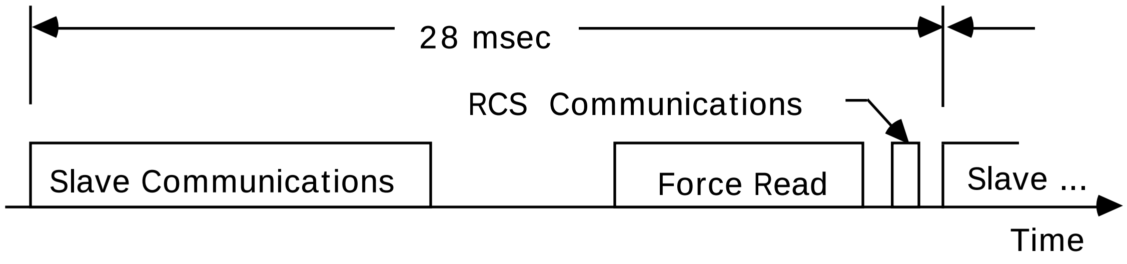


Figure 7. RCS Communication Timing.

The problem with this order is that VAL runs at its own rate and RCS must synchronize to it, not the other way around. To solve this problem, the communication process starts a timer when VAL starts communicating. The timer is then used to start the force sensor and to start the RCS communication.

The routine WAIT-TIME pauses until the timer is at the value wait-time.

2. Force Words

The routine `SEND-FORCE` reads the force sensor, nulls the readings, and stores the values in the system-parameter-file.

The routine `FIX-FORCE` monitors the force health bit. If there is an error, `FIX-FORCE` tries to clear the sensor by reading any remaining data points. If that does not clear the error, `FIX-FORCE` resets the sensor by sending ^X. To clear the sensor from the terminal, enter: **comm>** `INIT-FORCE.`

VIII. WORKSTATION INTERFACE

This chapter describes the interface between the SUN and the PUMA 760 at the Cleaning and Deburring Workstation. The chapter is divided into three parts: the interface to the RSL board; the interface to the Task board; and the communication protocol used by both boards.

1. RSL COMMAND INTERFACE

All lines sent by the SUN to the RSL board are executed as if they were typed at the terminal or loaded from a block. RSL or any recognizable commands can be sent. For example, `3 D>M` can be sent causing RSL to reset itself.

All values are sent in ASCII. String variables can be up to 15 characters long and can be comprised of any ASCII characters except `CR` and `SPACE`; however, only printable characters are recommended. Integers are 16 bits. Floating point numbers have two formats:

E format:
```
SPACE -123.45E-67 SPACE
```
Free format:
```
SPACE -123.456 SPACE
```

The decimal point must be shown.

Send the desired deburring paths to the RSL board. To redefine a loop with new values send a path with the same part name and loop number to RSL. To delete an individual loop or all the loops on a part send one of the following to RSL:

```
DELETE-LOOP   ["part-name"]  [loop#]

DELETE-PART   ["part-name"]
```

While Task level is executing, RSL can define new loops, redefine old loops, or remove old loops as long as the loops are not referenced by the commands sent to Task.

2. TASK COMMAND INTERFACE

There are two new commands to Task: `DEBURR` and `TEACH`. To command the robot to deburr one or more loops send the command:

```
DEBURR ["part-name"] [loop#] {[loop#]} ... {[loop#]} ;;
```

To run the self-correction routine, send the command

```
TEACH ["part-name"] [loop#] {[loop#]} ... {[loop#]} ;;
```

to the Task board. Currently only the end brush is taught. If loops with different tools are sent down, then they run as normal deburr loops.

All lines sent to the Task board are sent directly to the Task input command buffer. Thus any legal Task command can be sent. Such as:

```
RESTART
PAUSE
QC
JOYSTICK
GOTO [obj] [grip#] [loc type] [loc name]
```

See RSL documentation for specific parameters.

3. COMMUNICATION PROTOCOL

There are three RS-232 lines between the SUN workstation and the 760 RCS bucket. Line 1 is connected to the RSL board and lines 2 and 3 are connected to the Task Board. On line 1 the SUN sends -paths- and commands such as DELETE-LOOP to RSL. On line 2 the SUN sends commands to Task. On line 3 the SUN receives status from Task.

The protocol on the command lines 1 and 2 is:

The SUN sends `STX` one or more times.
When RCS echoes this, the SUN sends a complete line. A complete line can be the entire -path-, the -path- header, a complete -ppt-, or a command such as `REMOVE-LOOP`. The maximum line length is 1000 bytes.
RCS echoes each character.
The SUN checks the characters to see that they are the same.
If they are not correct, the SUN sends `STX`, waits for echo, and retransmits the line.
Once the line has been sent correctly, the SUN sends `SI ETX`.
RCS echoes these and begins processing the line.
When RCS is done and if there are no errors, RCS sends two `LF` and then one about every second until the SUN the sends `STX`.
If there was an error and RCS is still alive, RCS sends "`0`".

In HEX, `STX` = `02`, `SI` = `0F`, `ETX` = `03`, `LF` = `10`, and "`0`" = `30`.

The protocol on the status line 3 is:

Task sends a status byte each time the status changes and then sends the byte again about every second. The status bytes are:

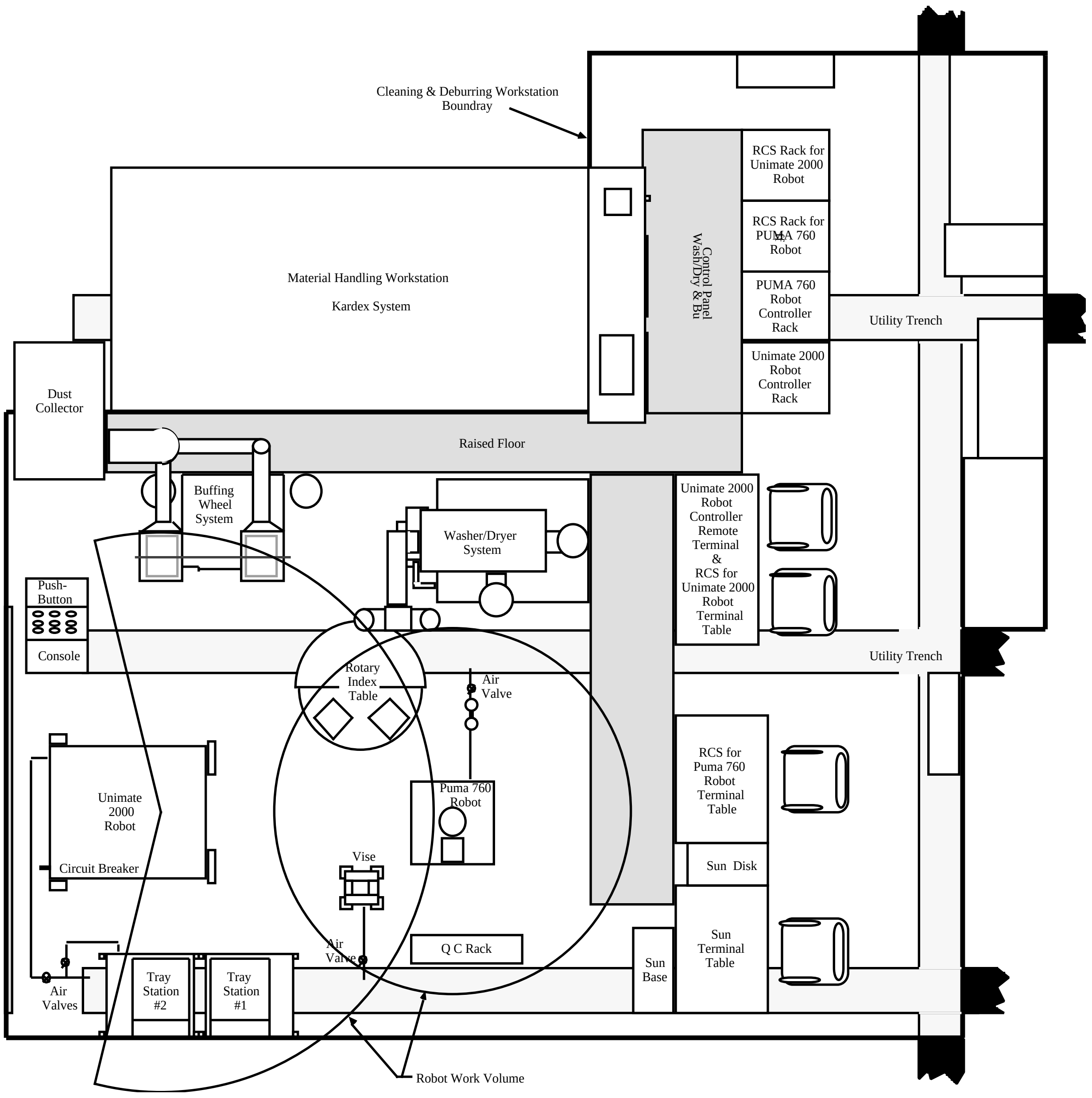
`01`	if Task is executing and has control of the vise,
`02`	if Task is executing and does not have control of the vise,
`04`	if Task is done and has control of the vise,
`07`	if Task is done and does not have control of the vise,
`[#]`	if Task has an error.

Where `[#]` is the error number plus `30` HEX. For example, error 7 would be sent as `37` HEX.

Hardware: 25 pin D-Plug, SUN Tx on pin 2, SUN Rx on pin 3, Gnd on pin 7.

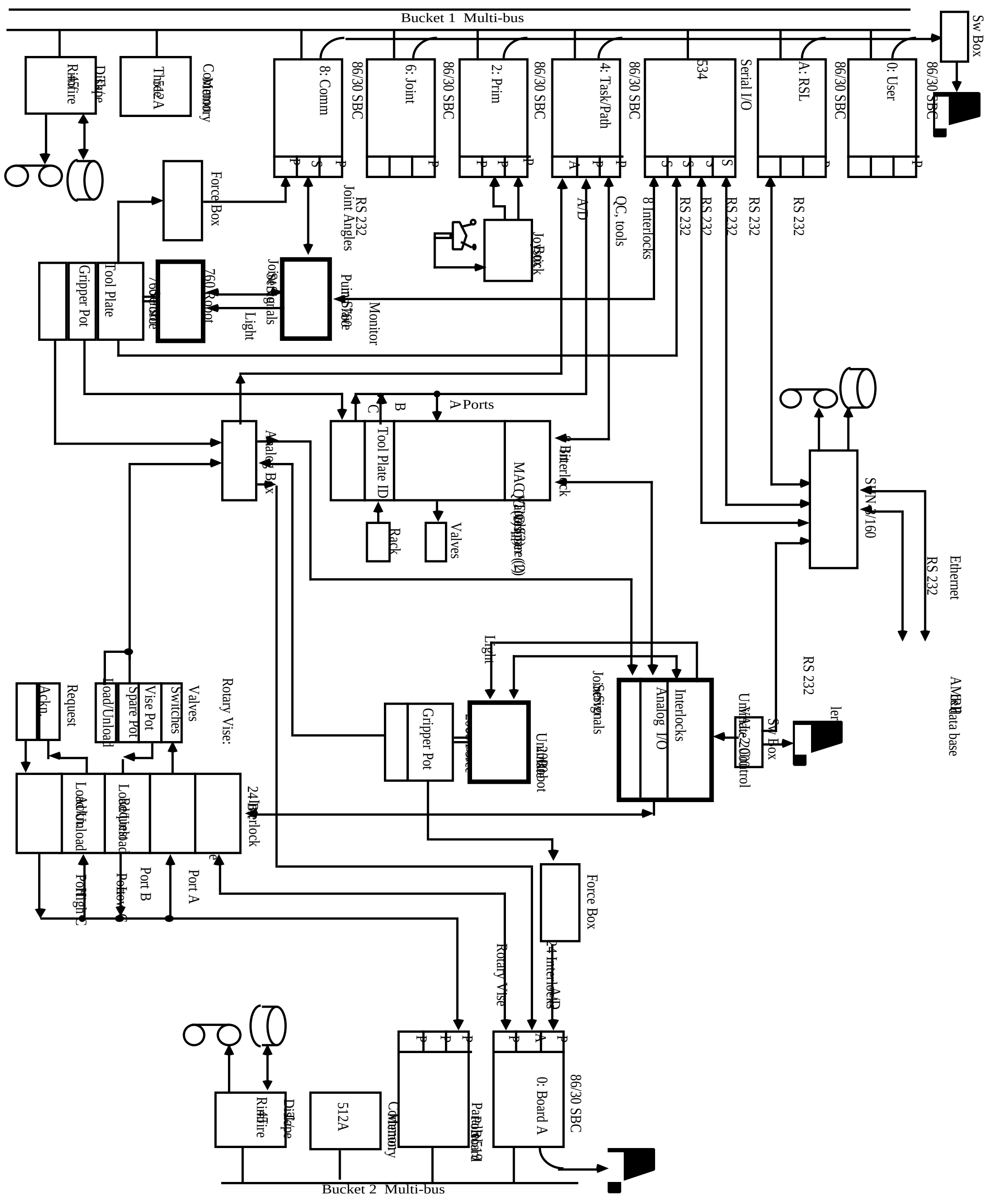
APPENDIX A: HARDWARE

1. WORKSTATION LAYOUT



Cleaning & Deburring Workstation Boundray
RCS Rack for Unimate 2000 Robot
RCS Rack for PUMA 760 Robot
PUMA 760 Robot Controller Rack
Unimate 2000 Robot Controller Rack
Control Panel Wash/Dry & Bu
Utility Trench
Material Handling Workstation
Kardex System
Dust Collector
Raised Floor
Buffing Wheel System
Washer/Dryer System
Unimate 2000 Robot Controller Remote Terminal & RCS for Unimate 2000 Robot Terminal Table
Push-Button
Console
Utility Trench
Rotary Index Table
Air Valve
RCS for Puma 760 Robot Terminal Table
Unimate 2000 Robot
Puma 760 Robot
Sun Disk
Circuit Breaker
Vise
Air Valve
Q C Rack
Sun Terminal Table
Sun Base
Air Valves
Tray Station #2
Tray Station #1
Robot Work Volume

2. WORKSTATION ELECTRICAL SCHEMATIC



3. MULTIBUS ADDRESSING

3.1 Multibus Memory Map

Page 0

0,0000	1,FFFF	Processor 0	master board	86/30
2,0000	3,FFFF	Processor 2	Prim	86/30
4,0000	5,FFFF	Processor 4	Task & Path	86/30
6,0000	7,FFFF	Processor 6	Joint	86/30
8,0000	9,FFFF	Processor 8	Comm	86/30
A,0000	B,FFFF	Processor A	RSL	86/30
C,0000	D,FFFF	Processor C	Spare	86/30
E,0000	E,FFFF	Common Memory bd 1		PSM 512 A
,0000	,EFFF	RCS Communication Buffer		
,F000	,FBFF	Free RAM		
,FC00	,FC1F	Disk Control		
,FC20	,FCFF	Software switch semaphore (need 2 bytes/board)		
,FD00	,FFFF	Free RAM		
F,0000	F,BFFF	Free Map Space		---
F,C000	F,FFFF	Proms		86/30 prom

Page 1

0,0000	1,FFFF	Not accessible (on board RAM)	
2,0000	7,FFFF	Common Memory Bd 2	PSM 512 A
8,0000	F,BFFF	Common Memory Bd 3	PSM 512 A
2,0000	4,FFFF	RSL File space = 3 segments	
5,0000	F,BFFF	System Dictionary = ~11 segments	
F,C000	F,FFFF	Not accessible (on board prom space)	

3.2 Multibus I/O Space

00	51	Free
52	53	Rimfire 45
54	5F	free
60	6F	534 Board (serial)
70	7F	free
80	9F	J4 multimodule on 86/30
A0	BF	J3 multimodule on 86/30
C0	DF	other 86/30 on board ports
E0	EF	free
F0	FF	free

3.3 Multibus Interrupts

3	SUN to Task command Rx from Port 1 on 534 board

3.4 On-board Interrupts

All Boards

0	6 msec I/O timeout
1	8087 error
3	on board serial port
4	on board serial port

RSL board

7	SUN Rx from serial multimodule

T&P board

2	RCS comm bit input
7	SUN to Task command from multi-bus interrupt 3

Prim and Joint boards

2	RCS comm bit input

Comm board

6	PUMA slave Tx	from serial multimodule
7	PUMA slave Rx	from serial multimodule

4. POWER ALLOCATION

	Count	A (max) at +5 V	A (max) at +12V	A (max) at -12V
8630 SBC w/ 8087	6	36.45	0.18	0.18
8630 SBC w/out 8087	0			
Rimfire 45 (disk control)	1	3.5		
PSM 512 (memory)	3	18.		
350 SBX (parallel MM)	3	0.96		
351 SBX (serial MM)*	3	1.38	0.09	0.09
311 SBX (A/D MM)	1	0.25	0.05	0.06
328 SBX (D/A MM)	0			
519 (parallel board)	0			
534 (serial board)	1	1.7	0.28	0.25
7438 paral. driver chip	2	0.09		
901 paral. terminator chip	19	1.43		
23 slot reset/pp circuit	1	0.4		
TOTAL		64.16	0.6	0.58

Notes:
1) 8630 boards have 32k proms.
2) Parallel port values are for buffer chips in the A port only.
   Add the driver and terminator chips up seperately.
3) Serial ports are in RS232C mode.
4) D/A MM are in voltage mode. For current loop mode, use 0.2A @ +12Volt / MM.
5) The following are estimates:
   8630 ±12V = 0.03A same as a 351 SBX
   PSM 512 = 6.0 A
   23 slot reset/pp = 0.4 A

APPENDIX B: OPERATION INSTRUCTIONS

This appendix explains how to start-up PUMA 760, shut-down the PUMA 760, and recover from basic errors. To operate other workstation equipment, see [7]. Before running any equipment, the operator should read Section 1, Workstation Safety.

1. WORKSTATION SAFETY

To operate workstation equipment, the operator should be fully trained in the workstation's safety procedures. Two of the more important procedures are:

- <u>WEAR SAFETY GOGGLES</u> or safety glasses with side shields at all times while at the workstation or on the shop floor. Safety goggles are available at all entrances to the shop floor. Ensure that all visitors wear the proper eye protection.

- <u>NEVER ENTER A ROBOT'S WORK VOLUME</u> when the arm power is on, signaled by red lights mounted on each robot. The robots' work volumes are marked by red painted floors and are delineated by black and yellow safety tape. Although the robots may appear to move in a predictable manor, a malfunctioning servo valve, for example, could cause the robot to move unexpectedly and at high speed.

<u>EMERGENCY STOP</u> There are two ways to stop the equipment in the workstation: HOLD and E-STOP. HOLD is a soft stop, used to temporarily stop the motion of one or both of the robots. When the HOLD is reset, the robot(s) continues the task as normal. E-STOP is a hard stop, used in emergency situations to turn off the power to all of the workstation equipment and apply brakes to the robot arms. When E-STOP is reset, the controllers will need to be initialized. An E-STOP is more difficult to recover from but provides a quicker and more reliable method of stopping the robots than HOLD does.

There are several stop boxes (See Figure 8.) spread about the workstation. Each has a large "E-STOP" button, which stops all workstation equipment; two "HOLD" buttons, one for the PUMA 760 Robot and one for the Unimate 2000 Robot; and three "RESET" buttons, one for the E-STOP, one for the PUMA HOLD and one for the Unimate HOLD. The stop boxes are wired in series so you can press "HOLD" on one stop box and "RESET" on another. There are also "HOLD" and "E-STOP" buttons on the PUMA 760 joystick but they only stop the PUMA 760 and do not stop any other equipment.

To 'HOLD' a robot, press the appropriate "HOLD" button on the stop box. It is a good idea to keep your hand near the "HOLD" button when you are issuing new commands. To recover from a "HOLD", press the appropriate "RESET" button, and the robot will continue to move.

To 'E-STOP' the equipment, press the palm-sized red "EMRG STOP" button. This turns off both robots' arm power, the pneumatic deburring tools, the rotary vise, the washer/dryer index table, the water pump, the heater, the buffing wheels, the dust collector, and the roller tray stations.

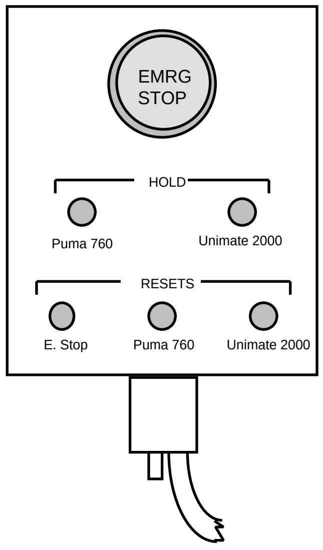


Figure 8. Emergency Stop Box

2. START-UP

*** Warning: **Do not initialize** the PUMA 760. This will erase all VAL programs and locations. Just say `N`.

1) Turn on the PUMA 760 Robot Controller Rack (the big red switch in back).

2) If the controller is off, hold AUTO START button in while you turn on the controller chassis. Continue holding the AUTO START button until you hear a click, about 3 sec.

   If the controller is already on, a) press the ARM POWER OFF button and then b) hold AUTO START button in while you toggle the LSI INIT switch (on the top of the chassis). Continue holding the AUTO START button until you hear a click, about 3 sec.

3) Clear people from the robot work volume.

4) Press ARM POWER button.

5) Press the JOINT button on the PUMA joystick and move the robot to the vise safe location. (The tool is vertical, the tool tip is midway between the quick change rack and the robot base, and the tool tip is slightly higher than the top of the quick change rack. See Figure. 9.)

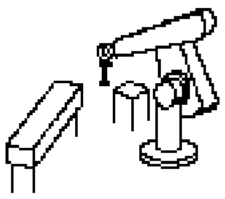


Figure 9. Vise safe location.

*** Warning: The robot joints will move about 10 degrees during the following step. Keep a hand near the ARM POWER OFF or E-STOP button.

6) Press the COMP button on the PUMA joystick. The robot will move as it calibrates itself.

7) Set the terminal switchbox to RCS, if not already there.

8) Turn on: the 760 RCS rack, all the chassis in the rack, and the RCS terminal.

9) Press the black "SYS Reset" button on the terminal switch box. If not already set, switch the black rotary switch to MST, and set the toggle switch to TERM.

*** Warning: The robot will move to the vise safe location during the following step. Make sure it is free to move, or keep your hand on the HOLD SET button.

10) Press function keys `f9` then `f10`. Or, if the terminal was off, enter:

```
HEX F7 C9 OUTPUT 26F0 30 ERASE 0 CBOOT 0 MBOOT 4 D>M
```

After the disk spins up, the computer will begin to print messages from the various boards. The robot will move to the vise safe location. The tip of the robot tool should be higher than the top of the quick change rack. If not see errors below. You should end up on **`Mst>`** with the message: **`*** System Running ***  DONE`**.

The 760 RCS system is now running and ready to receive commands from the SUN.

3. SHUT-DOWN:

1) **`Mst>`** `BYE`. This stops the levels, unloads the disk, and starts "snake" (the screen saver).

2) Move arm to a safe location, press the ARM POWER OFF button.

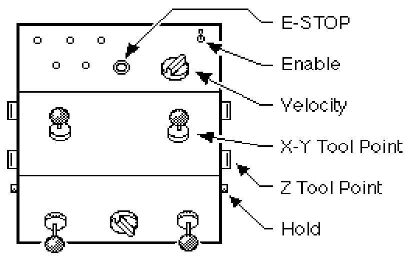
3) Turn off all equipment that you turned on.

4. GENERAL INFORMATION:

MOVING FROM BOARD TO BOARD: The RCS system has 6 boards. There is a master board, **`Mst>`**, and 5 slave boards, **`rsl>`**, **`t&p>`** (Task and Path), **`prim>`**, **`joint>`** and **`comm>`**. You must enter different things on different boards. In these directions, the prompt tells you the board to enter commands on. For example: **`Mst>`** `GO`. means type a `G`, an `O`, and `RETURN` on the **`Mst>`** board. On the computer screen, the prompt shows the name of the board you are on. Some commands to the master board send commands to slave boards, ex: `Mst> GO`.

To move from board to board use the function keys. You can move back and forth between the master board and any slave board. To move between two slave boards, you must move to the master board first. The RCS terminal has 11 function keys labeled `f1` thru `f11`. Each has a separate function as labled on the duct tape. For example, `f3` will move you to the `Mst>` board.

JOYSTICK: Enter either `Mst> JOYSTICK` or `t&p> 912 LOAD`. These commands clear certain errors and enable the joystick. To move the robot, flip the enable switch up and set the velocity to a low value. (Rotate clockwise.) The most common motions are tool X, tool Y and tool Z.



STATUS: Enter `t&p> STAT`. The status of the robot will be printed.

COMMAND DIRECTORIES: `t&p> 910 LIST` and `t&p> 920 LIST`. These blocks list different commands to the robot. Ex: `t&p> 924 LOAD`. moves to VISE-SAFE.

SUN COMMANDS: `t&p> ENABLE-SUN-CMD` and `t&p> DISABLE-SUN-CMD` tells Task level to accept/ignore commands from the workstation controller.

VISE: `t&p> GET-VISE` locks the 2000 out of the vise volume. `t&p> DROP-VISE` drops the vise volume request. The 2000 can now get the vise.

RSL: At start-up, the RSL board is told to accept robot paths from the SUN. This program is called: `rsl> BGO`. If RSL is to ignore the paths from the SUN enter: `Mst> ABT_RSL` `rsl> DGO`. You will not get a prompt back. To abort these processes, `BGO` or `DGO`, enter: `Mst> ABT RSL`.

5. ERRORS:

Most errors can be corrected by doing step (a). If the joystick is needed, see Section 4.

a) Enter: `Mst> GO`. Then, if there is no error, enable joystick and move the robot near the vise safe location. On the Task board enter: `t&p> 924 LOAD`. The robot will move to the vise safe location. The tip of the robot tool should be higher than the top of the quick change rack. If not see error (d) below.

b) Force sensor is down.

- **`comm>`** `INIT-FORCE`.
- Check power to sensor chassis, cable connections, etc.

c) The robot does not move
- It may already be at vise-safe. Try **`t&p>`** `922 LOAD`. The robot should move to qc-gate.
- Press the HOLD CLEAR button on the RCS joystick.
- See General errors

d) The robot does not move to vise-safe location, the tool is lower than the top of the quick change. This generally means that the robot does not know that it has a tool.
- **`t&p>`** `911 LOAD`, commands reset, then **`t&p>`** `924 LOAD`.
- Check power to bucket-1 interface chassis, cable connections, etc.

e) Quick Change Error:
If the robot has an error over the quick change and stops moving:
`t&p>` `X-UNLOCK` **`Unlock quick change?`** `Y`
t&p> 911 LOAD. This restarts RCS. **`t&p>`** `912 LOAD`. This activates the joystick

f) No air: tools don't work, quick change doesn't work
- Turn on air valve feeding the PUMA 760 robot.

g) No Auto Start program.
- **Do not initialize!**
- Do steps 1-5 above without pressing auto start button. The robot will not move.
- Set terminal switch box to LOCAL and TERM.
- Enter `CAL`. Answer `Y`. The robot will move.
- Do step 7 above.

LIST OF REFERENCES

[1] "The NBS Real-time Control System / User Reference Manual". Editors: Leake, S. and Kilmer, R. NBS Technical Note 1250. Aug. 1988.

[2] Murphy, K., Norcross, R., and Proctor, F., "CAD Directed Robotic Deburring", 2nd International Symposium on Robotics and Manufacturing Research, Education, and Applications, Albuquerque, NM, 1988.

[3] Norcross, R.,"CDWS Workstation Control Structure", IEEE International Conference on Robotics and Automation, Philadelphia, PA, 1988.

[4] Norcross, R., "CDWS Workstation Controller Reference Manual", National Bureau of Standards, NBS Technical Note, to be published.

[5] Programming Manual, User's Guide to VAL-II, Version 2.0, 398AG1, Unimation, December 1986.

[6] Proctor, F., "CAD Directed Automatic Part Handling", National Bureau of Standards, NBS Technical Note, to be published.

[7] Murphy, K., Norcross, R., Tanguy, P., and Proctor, F., "Cleaning and Deburring Workstation Operations Manual", National Bureau of Standards Internal Report 88-3804, June 9, 1988.