NISTIR 88-3890

THE VERTICAL MACHINING WORKSTATION SYSTEMS

November 3, 1988

By:
Jau-Shi Jun

National Institute of Standards and Technology
formerly
U.S. DEPARTMENT OF COMMERCE National Bureau of Standards Gaithersburg, Maryland

THE VERTICAL MACHINING WORKSTATION SYSTEMS

Dr. Jau-Shi (Jay) Jun
Computer Scientist
Factory Automation Systems Division
Center for Manufacturing Engineering
National Institute of Standards and Technology

November 3, 1988

Certain commercial equipment and software are identified in this paper in order to adequately specify the experimental facility. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the equipment or software identified are necessarily the best available for the purpose.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION TO THIS MANUAL

1. PURPOSE OF THIS DOCUMENT

This paper presents an overview of the control systems of the Vertical Machining Workstation (VWS) of the Automated Manufacturing Research Facility (AMRF) at the National Institute of Standards and Technology (NIST). It also gives details of workstation control flow. Refer to other documents for details of other VWS systems.

2. AUDIENCE

This paper is intended to be useful to people interested in concepts and technical details of the VWS, particularly AMRF personnel who are running the VWS or maintaining or improving the software for the VWS. The paper is intended to be useful also to other researchers in automated manufacturing.

II. SYSTEM OVERVIEW

The Vertical Machining Workstation (VWS) is the two lowest level of control in the Automated Manufacturing Research Facility (AMRF) at the National Institute of Standards and Technology. The five levels of the control hierarchy are the Facility level, the Shop level, the Cell level, the Workstation controller level, and the Equipment level. Figure 1 shows the VWS in the AMRF control hierarchy. Refer to other documents for the AMRF architecture description. [Ref1] [Ref2]

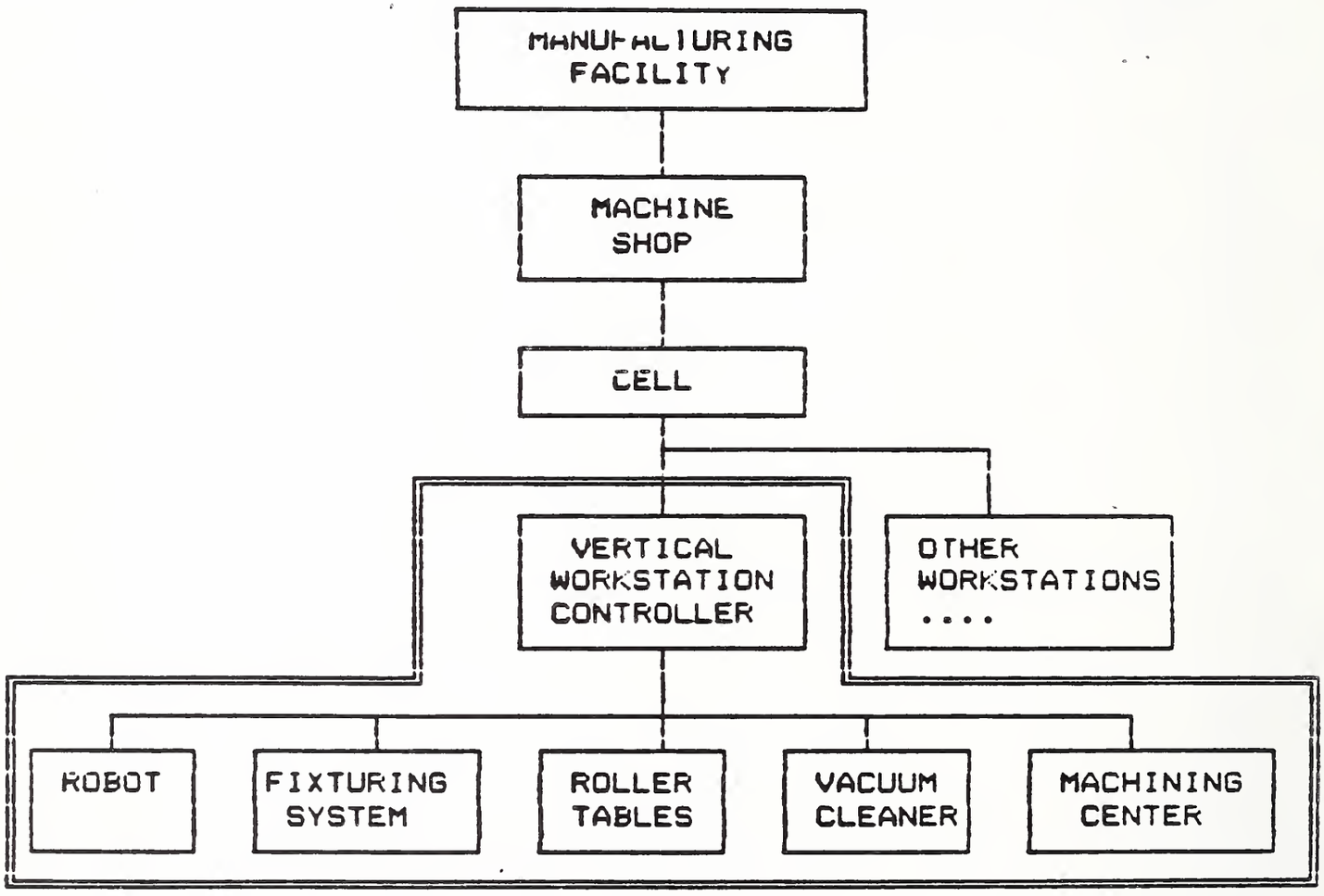


Fig.1 VWS in the AMRF Control Hierarchy

At the workstation level of control, the VWS is responsible for carrying out process plans for machining raw material to a finished part by coordinating the activities of several pieces of

equipment (the lowest level in the AMRF control hierarchy): a vertical machining center, a robot with gripper system, a vacuum cleaner-based chip removal system, two fixture devices, and two tray roller tables.

The VWS normally functions under the control of the Cell. This is called the integrated mode in this document. It can also function as a stand-alone system under the control of an operator (called stand-alone mode).

To operate the system, different types of instruction lists are needed for different levels of control hierarchy and equipment. These instruction lists are called process plans. The process plans are prepared by the Process Planning System. Capabilities for creating the plans within the VWS are also provided.

In this document, we will discuss the objective of the VWS system design, the software architecture of the VWS controller, the way the system is implemented, and how the Vertical Workstation is operated. We will also discuss what kinds of data are expected by the VWS, how the data are prepared, and how data are used by the system. Individual software packages are discussed in detail at the end. System programmers will find that portion of the document useful.

Detailed discussion of process plan preparation is given elsewhere [Ref12] [Ref9] [Ref10]. The description of the equipment-level components, hardware and software, may be found in separate papers. [Ref3] [Ref18]

III. GENERAL DESCRIPTION OF THE CONTROL SYSTEM OF THE VWS

1. BACKGROUND

The VWS controller project was started in early 1985. The first demonstrable system was completed later that year. The system has gone through two major revisions since then. The earliest prototype was a stand-alone system. The second system had the VWS running in the integrated mode and was demonstrated in late 1986.[Ref5] In the integrated mode, the VWS runs under the control of the Cell [Ref4] via the AMRF network [Ref7] and uses the IMDAS (Integrated Manufacturing Database Administration System), the distributed data system which provides common interfaces to the AMRF's user programs and underlying databases.

The major goals in implementing the VWS were that the system must:

Be easy to use,
Be as automatic as possible,
Be as generic as possible, and
Make effective use of resources.

To meet the last goal, the VWS can run two jobs at the same time, one foreground job and one background job. While the machining center is busy cutting a part (foreground job), the robot can work on a background job, such as changing the robot fingers, or moving a part from one tray to another.

In the VWS, the numerical control (NC) code used by the machine tool is generated dynamically from a part design and its process plan. This is analogous to the use of a high-level language for computer programming which removes the hardware dependency. The process plans for the milling machine may be prepared by the AMRF Process Planning System. Process plans for the machining center can also be automatically prepared at the VWS by using the Process Planner. The Process Planner uses the part design data (output from the Part Design System) to generate the process plan. Both the Process Planner[Ref12] and the Part Design System[Ref11] were developed specifically for the VWS (more discussion later).

To secure a part for the machining process or move a machined part to an output tray, the VWS' equipment programs coordinate the robot, the gripper system, the roller tables, the fixturing

system, and the finger exchange system. These programs are generated dynamically by the VWS when needed.[1] [Ref6]

When the VWS runs in the stand-alone mode, a part can be designed and processed (cut) within two hours.

2. WHAT DOES THE VWS CONSIST OF ?

Figure 2 shows the floor layout of the physical components of the VWS. Figure 3 shows the software components and their relationships.

2.1. Hardware Components

2.1.1. Computer Housing the VWS Controller

The computer used for the VWS controller is a Sun Microsystems' workstation model 3/160. The computer has 6M bytes of RAM and shares a 350M-byte disk storage file server with other Sun computers. The Sun computer has a large screen with graphics capability and window tools. Six windows are used to display menus, system status, and diagnostic messages. A graphics window is used to display the part design. The Sun computer communicates with other Sun computers via an EtherNet interface, which, in turn, communicates with other computers used in the AMRF via the AMRF network. The VWS controller communicates with its subordinate, the Central Equipment Controller (CEC, also called Equipment Interface Processor), via an RS-232 interface.

2.1.2. Central Equipment Controller (HP-9000 computer)

An HP-9000 is directly connected to the robot controller, the gripper system, the GE-2000 machine tool controller which controls the vertical machining center, the roller tables, and the vacuum cleaning system. All connections to the HP-9000 are via RS-232 interfaces.

2.1.3. Vertical Machining Center

The Monarch VMC 75 is a numerically controlled (NC) vertical machining center. It is controlled by a GE 2000 controller.

---

[1] The equipment control programs may be pre-generated, visually inspected, and stored in the control program library for later use.

Refer to [Ref3] [Ref18], the VWS hardware components description for more details.

2.1.4. Robot

A Unimate 4070 series robot is used for transferring the material between the part buffer and the fixture. An in-house developed gripper system with interchangeable fingers is used to pick up parts of different sizes and shapes.

2.1.5. Chip Removal System

Two approaches have been used to clear the chips accumulated during the machining process. The first approach employed a 2.5 horsepower (American Vacuum Co. Model PB-15) vacuum cleaner. The robot, with a specially designed gripper, picked up a flexible vacuum hose and cleaned the chips from the top of the machined part. However there were problems in getting the robot to clean all of the chips effectively.

Recently, a different approach has been used to clean the chips. A brush is installed in one of the tool slots. And at the appropriate time, a special chip cleaning NC program is run which uses the brush to clean chips off the vise or the top of the machined part. The method seems to be more reliable than the other, but further development work is needed in this area.

2.1.6. Fixture Devices

Two fixture devices, a pneumatic vise and a hydraulic pallet mounting system, are used to hold parts of different sizes and shapes. Both devices were developed in-house.

2.1.7. Part Buffers

Two roller tables are used to receive or ship trays. These roller tables may be controlled either by the VWS or the Material Handling System (MHS). Both software semaphores and hardware locking mechanisms are used to avoid simultaneous attempts to use a tray.

2.1.8. Two Robot Finger Changers

The VWS has two robot finger changing stands developed in house. The robot uses these two finger changing stands to mount and

dismount different sets of fingers. The robot end effector attaches to one set or the other for picking up parts of different shapes and sizes.

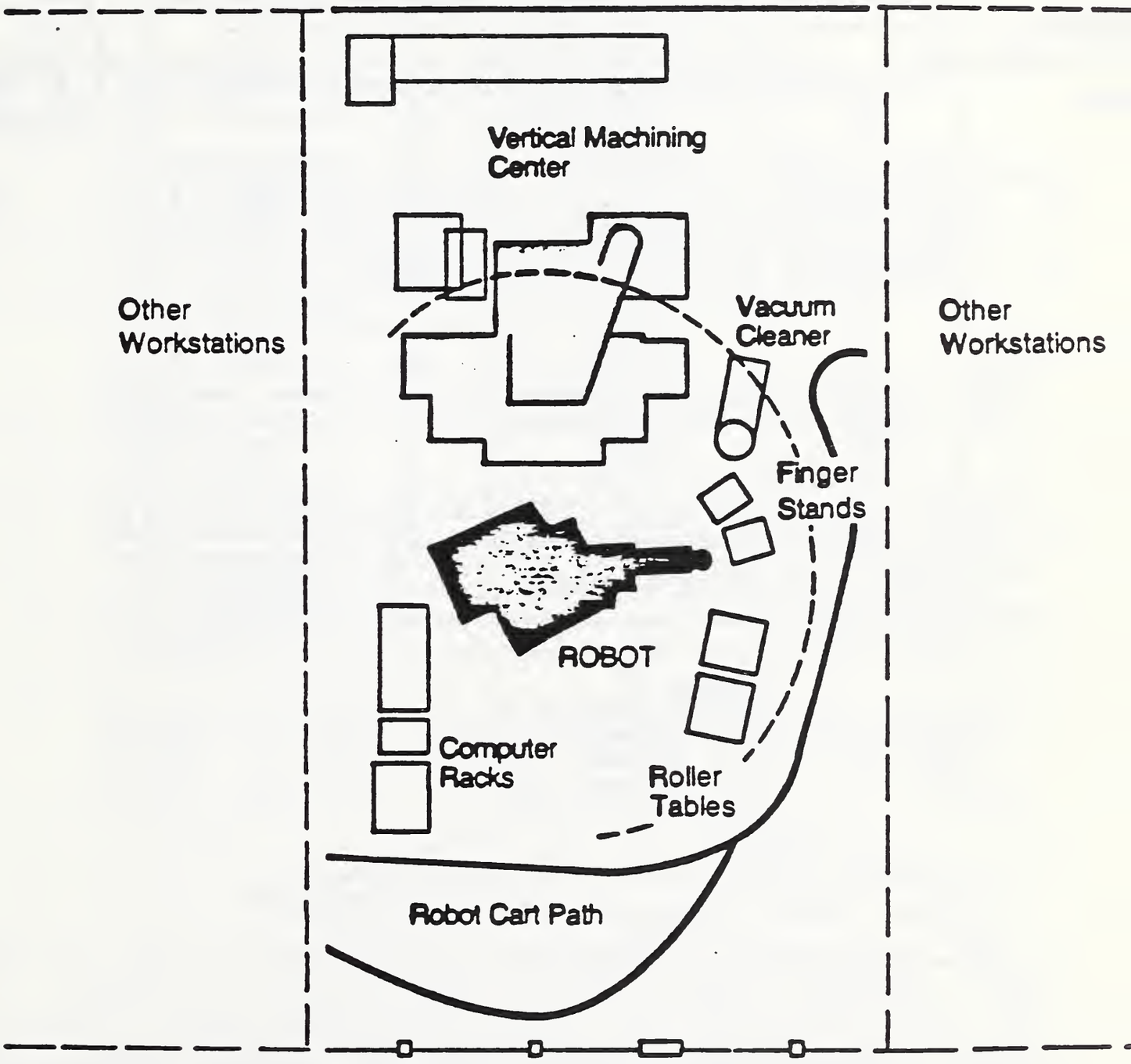


Fig.2 Floor Layout of the Physical Components of the VWS.

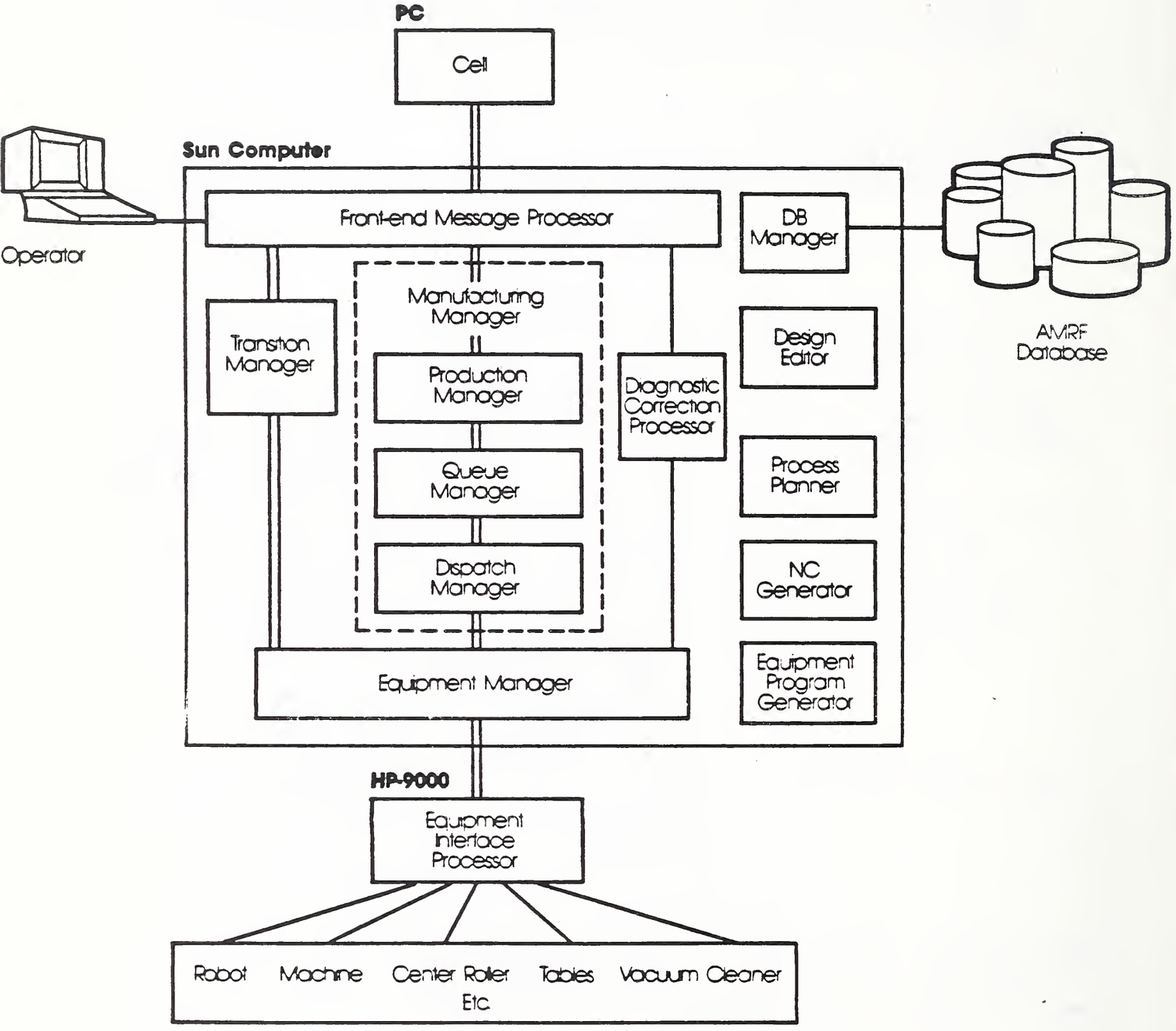


Fig.3 VWS Control Software Components and Their Relations.

2.2. Software

2.2.1. User Interface

Special attention is paid to the user interface area in the VWS workstation implementation. Graphic displays, menus, and mouse functions are used to help an operator control the workstation.

2.2.2. Types of Usage

There are two types of usage in the VWS workstation: using the VWS as a manufacturing controller or using the VWS as a manufacturing data preparation station.

* Manufacturing Controller: The VWS controller coordinates its subordinate equipment to produce a finished part. In this usage, the VWS may be a stand-alone workstation controlled by an operator, or it may be controlled by the Cell controller functioning in the AMRF integrated operation mode.

  Six windows are used to display various types of system statuses, control menus, and diagnostic messages. One graphic window is used to display the part design, if applicable.

* Data Preparation Workstation: By means of its feature-based design system, the VWS is used to design a part, to create process plans and NC code, to generate equipment programs that coordinate robot, vise, roller tables, etc., and to create state-table logic for the controller software.

  The user interface to the system is based on a series of prompts and menus, plus a graphic display of the part design. A mouse device is used for part of the data input. This makes it easy for a novice user to design complex parts.

2.2.3. The Language Used

The VWS controller software is implemented in Franz LISP, with the exception of a few routines written in the C language. LISP is a symbolic programming language that is used extensively for artificial intelligence applications. The LISP environment (the whole control system) runs under the Sun's UNIX operating system. Parts of the communications software package are written in C. The software running on the HP-9000, which is used as the equipment interface processor, is written in HP BASIC.

2.2.4. Software Packages Grouped by Functions

Functionally, The VWS controller software falls into the following groups:

* Production Management Operation System (PMOS)
* Front-end Message Processor (FMP)
* Manufacturing Manager (MM)
* Transition Manager (TM)
* Equipment Manager (EM)
* Equipment Command Verifier (ECV)
* Test and Maintenance Manager
* Communication Manager
* Database Manager
* Data Preparation
  - Automated Equipment Program Generator (EPG)
  - State-table Preparation System
  - Feature-Based Part Design System
  - Automated Process Planner
  - Automated NC Generator
  - Workstation level Process Plan Editor

2.2.4.1. Production Management Operation System (PMOS)

The PMOS is the backbone of the control system. It schedules, monitors and executes a set of Finite State Machine Modules (FSM) [Ref20] [Ref21] just as other operating systems execute a set of application programs. It provides a rich repertoire of commands for real-time control by an operator from the console keyboard. Taking advantage of the large screen and the powerful window system provided by the Sun computer, a total of 6 windows are used for controlling the VWS and displaying the control system statuses. System status can be selectively displayed, logged to a file, or both in real-time.

The PMOS logically executes a set of FSM's in parallel in each cycle. This is different from most operating systems, which run a set of application programs in a sequential manner. In the PMOS system, all inputs/outputs are done in a time-slice between cycles. This prevents synchronization problems where modules may execute asynchronously. Refer to the module description later in this document for more information.

2.2.4.2. Front-end Message Processor (FMP)

Depending on the mode of control operation, i.e. stand-alone or integrated with Cell control, the Front-end Message Processor (FMP) accepts input from either an operator from the console or

remotely from the Cell. When the VWS is operating under Cell control, the FMP allows an operator to gain control with pre-emptive commands from the console. Based on the command message, the FMP directs the relevant task to either the Transition Manager (TM, to be discussed later) or the Manufacturing Manager (MM, to be discussed next) for further processing.

2.2.4.3. Manufacturing Manager (MM)

The Manufacturing Manager is responsible for managing jobs that are part production related, such as: produce a part (MACHINE_LOT), receive a tray (RECEIVE_TRAY), ship a tray (SHIP_TRAY), setup a work area, etc. The current implementation supports the first three types of jobs just mentioned.

When the Manufacturing Manager (MM) gets a job from the FMP, the MM checks the parameters, retrieves relevant data from the database (either local or global database), and processes the job accordingly. Descriptions of three of MM jobs are given below.

* RECEIVE_TRAY - With this job order, the Cell informs the VWS that a tray with certain part blanks in it has been delivered to a roller table. The MM will retrieve the tray type, the tray contents, and the part blank descriptions from the database. The internal world data model is then updated with the following facts:

  - which roller table has the tray,
  - the tray's serial number and its type,
  - the part blanks' ids, their location in the tray, part orientation, and sizes.

  The above information is important for the automated equipment program generator and the equipment command verifier.

* MACHINE_LOT - one of the parameters of this job is an operation sheet (a workstation level process plan). The operation sheet is retrieved and scanned for additional preparation actions. It specifies a set of tasks to be carried out by the VWS. The MM then queues up these tasks, and dispatches one task at a time to the Equipment Manager (EM) to carry out the operation. The operation sheet is discussed more fully in Section IV.1.2. Each task may be executed by a single piece of equipment, such as the milling machine, or may involve a group of pieces of equipment, such as the robot, gripper, roller table, and vise working together to carry out a "ROBOT_LOAD_PART" operation.

* SHIP_TRAY - The Cell sends a SHIP_TRAY job order to inform the VWS to relinquish its control of the roller table, so that the MHS's robot cart may retrieve the tray and deliver it elsewhere. Normally, the MM receives this SHIP_TRAY job order after the job MACHINE_LOT has been completed, the finished workpiece has been placed back on the same location of the same tray as it came in, and a DONE status has been reported to the Cell.

2.2.4.4. Transition Manager (TM)

The Transition Manager is responsible for synchronizing transitions between the controllers in the AMRF hierarchy. Typical transitions include synchronization, warm startup, cold shutdown, restart, reconfiguration, and changing data or control modes. This framework, referred to as the UVA protocol, is defined and presented by Reynolds and O'Halloran[Ref14] and is partially implemented by O'Halloran for the VWS. It is in the form of a state-table module under the control of PMOS.

The TM is responsible for startup, shutdown, and restart tasks. In the future, it will also manage aspects of error recovery. The TM also coordinates change of mode of operation from stand-alone mode to integrated mode, or vice versa.

2.2.4.5. Equipment Manager (EM)

The Equipment Manager is at the lowest level in the VWS control hierarchy. It accepts commands from the Transition Manager (TM) as well as the Manufacturing Manager (MM). Each command represents a task of the MM or the TM, and is considered as a job from the EM's point of view. Included with each job (or command) is a job description which is in the form of attribute-name and attribute-value pairs. The EM decomposes the job into tasks based on the job nature and its job description. The job decomposition process could be simple or complicated, but it is totally automated. This is the unique part of the VWS implementation. After the job decomposition, the job becomes a list of tasks. Depending on the complexity of the job, the number of tasks may be anywhere from one to hundreds.

With this list of tasks, the EM sends tasks, one at a time following the list order, to the CEC (HP-9000). The HP-9000 sends each task to the intended equipment to carry out the action. Upon completion of a task, a status is reported back to the EM, and the next task in the list then follows. This process continues until the EM sends all the tasks in the list.

For reasons of safety, before sending a task, the Equipment Command Verifier (ECV) is called to check whether it is safe to carry out the task. The checking process will be described in the ECV section.

The following are a few examples of tasks and how the EM processes them:

* VISE_OPEN or VISE_CLOSE - To open the vise or to close the vise. Both tasks are simple and need no further decomposition. They are sent to the CEC (HP-9000) as is.

* GENERATE_NC - To converts a process plan into the NC format usable by the GE-2000. If the work file specified by the work plan is in process plan format, the EM calls the NC Generator to generate the NC. If the work file is a feature-based part design document, the EM calls the Process Planner to create a process plan, and then calls the NC Generator to generate the NC.

* DOWNLOAD_NC - To download NC into GE-2000. Came with this task is a work plan which has an NC file name. This task initiates a file transfer activity between the VWS controller and the HP-9000 first, and a second file transfer activity from HP-9000 to the GE-2000.

* EXECUTE_NC - To produce a part with the specified NC. This task comes with a work plan which has an NC file name. The EM generates two subtasks for the milling machine tool for this task: "SELECT_NC" and "CYCLE_START". The first subtask "SELECT_NC" instructs the GE-2000 controller to select the named NC code from its memory. The second subtask "CYCLE_START" starts the machine tool operations.

* ROBOT_LOAD_PART - To instruct the robot to move a part into the fixture system, and to secure it properly for the machine tool operations. The parameters of this task specify the part identifier and fixture identifier. The Equipment Manager calls the Equipment Program Generator (EPG) to generate the control procedure. The equipment control procedure (program) generation is automatic, and in this case the number of steps in the control procedure is about 40. The EM then uses this procedure. One step at a time is sent to the equipment interface controller for controlling the equipment. For reasons of safety, each step is checked by the Equipment Command Verifier (ECV) before sending. For each step (command) sent down to the HP-9000, status is reported back. With this

reported status, the EM calls the Internal Data Model Manager to update the table contains the VWS's current statuses. This table is referred to as the VWS internal world model in this document.

* If a task comes from the Transition Manager to shut down a device, the Equipment Manager first marks the device unavailable, sends a device shutdown command to that device, and updates the database to release the device. (Only partially implemented.)

2.2.4.6. Equipment Program Generator (EPG)

An Equipment Program Generator (EPG) is used to generate the equipment control procedures. The EPG is called by the EM if the task received is related to part loading, part unloading, part repositioning, robot finger changing etc..

Using information from the internal world data model, the task and its parameters, the EPG generates control sequences for the subordinate equipment. A detailed discussion of how the program is generated and what the program looks like are in the description of "THE VWS SOFTWARE PACKAGES" section (VI. 2. 2.2) later in this document.

An example is in order. For the task (ROBOT_LOAD_PART) and its three parameters (attribute-values pairs):

```
ROBOT_LOAD_PART
        WORK_PLAN       LOAD12 (optional)
        PART            PART#12
        TO              VISE,
```

the EPG generates the equipment program to do the following operations.

* lock the roller table
* change robot finger if necessary
* move robot to pickup PART#12 from its current position
* position robot with part above the vise
* open the vise if necessary
* load the part into the vise
* move the robot back to its home position
* close the vise

An example of a generated equipment program is shown in APPENDIX-5.

2.2.4.7. Equipment Command Verifier (ECV)

The ECV is used for verifying whether it is safe to execute an equipment command.

Before sending a command to the CEC, the EM calls the ECV to examine the command. Any command judged unsafe by the ECV will NOT be executed. And the system will switch to a manual mode waiting for operator intervention.

The ECV verifies one command at a time, it DOES NOT keep history. Using the data in the internal data model (VWS-World-model) at that moment and a set of safety rules, the ECV examines the command for that circumstance. For example, to have the robot place a held part in the vise while the vise is closed would not be allowed. The current implementation is still incomplete.

2.2.4.8. Test and Maintenance Manager

The Test and Maintenance Manager gains control when an error or an abnormal situation is encountered (based on analysis of equipment status received from the HP processor). If the error is correctable, the VWS tries to correct it; if it is uncorrectable, this information is passed up the control chain to the operator or the Cell. In the current implementation, for reasons of safety, the diagnostic correction processor always passes control to the operator. The operator interface provides means to bypass a problem, to retry an unsuccessful operation, or to abort an operation.

Various levels of system information and an events trace may be displayed on the screen or logged into files by an operator command.

2.2.4.9. Communication System

Communication between the VWS and its subordinate equipment currently is based on a serial interface. Communication between the VWS and the Cell or the IMDAS is through messages placed in the common memory areas. The AMRF network software reads and writes messages in the common memory area and transfers data to and from the IMDAS or Cell. Refer to the AMRF network document for details.[Ref7]

2.2.4.10. Data Management System

The Data Management System provides services to both the external database - IMDAS [Ref15] and the internal "VWS-World-Model". [Ref8]

2.2.4.11. State-table Editor

The State-table Editor (STE) is an editor specially implemented for preparing or editing the state-table modules. The program coding of a state-table module involves a lot of column and row manipulations. The STE displays modules in spread sheet form and has a rich set of instructions to manipulate columns and rows. The STE makes the job of preparing a state-table module much easier.

2.2.4.12. Data Preparation

2.2.4.12.1. Feature-Based Part Design Editor

A feature-based Part Design Editor (PDE) provides a stand-alone computer aided design (CAD) capability for the generation of part designs. A separate document specifically describes this capability.[Ref11]

2.2.4.12.2. Automated Process Planner

A Automated Process Planner is used to generate a process plan for the VWS vertical machining center. The Automated Process Planner is called by the VWS controller at run-time or is initiated by an operator in off-line mode to generate a process plan for the machine tool. A separate document specifically describes this capability.[Ref12]

2.2.4.12.3. Automated NC Generator

An NC code Generator is used to prepare NC code from the process plan and its corresponding part design document. The NC generation process may be initiated off-line by running the VWS Data Execution Module[Ref13], and the generated NC may be saved for later use. The NC Generator may also be called by the system to dynamically generate NC.

3. HOW DO YOU RUN THE VWS?

The VWS is a self-sufficient system; it can run by itself in "stand-alone mode". In this mode, an operator is in control. When the VWS runs under the AMRF systems, called "integrated mode", it is under the control of the Cell. Yet another mode is called "Test and Maintenance Mode", in which an operator can control the equipment step by step, or task by task. A discussion of how to run the VWS in each mode follows.

APPENDIX-6 is an example of the VWS operation screen.

3.1. Stand-alone Mode

The following steps are the instructions to set up and operate the VWS in stand-alone mode.

(1). Power up all the VWS equipment
(2). Turn the Sun computer on
(3). Login as user: **vws**
(4). Enter **vc** (for **v**ws **c**ontrol)
(5). The VWS goes through the station cold boot procedure, sets up the windows, and displays instructions for further operator control.
(6). Type **lcm** (local control mode) after seeing the operation mode selection message. The VWS is to be controlled by an operator from the console keyboard.

The system may use either the local database or the IMDAS. The default database mode is the local database mode. The system is now ready to use the local database to make parts. The two database modes may be switched back and forth with the following commands:

* **rdm** (remote data mode) - If the IMDAS is to be used, enter this command. This command works only if the IMDAS is up.

* **ldm** (local data mode) - The user may switch to the local database system by entering this command. Notice, this command is used only after an **rdm** command has been issued. The **ldm** is to undo whatever the **rdm** has done.

The status window will display the fact that the manufacturing manager is READY, the cmode (command mode) is LOCAL, and the dmode (data mode) is LOCAL or REMOTE. At this point, the VWS is

ready to accept manufacturing related commands. Three commands are supported:

* **rt** - RECEIVE_TRAY
* **m** - MACHINE_LOT
* **st** - SHIP_TRAY

For each command entered, the operator will be prompted with questions for further inputs.

In the case of a **receive_tray** command, the prompt will be:

```
Enter RECEIVE_TRAY cmd file ?
```

The following receive tray cmd files are supported now:

```
rt200_1          TRAY_200  at  ROLLER_TABLE_1
rt200_2          TRAY_200  at  ROLLER_TABLE_2
rt220_1          TRAY_220  at  ROLLER_TABLE_1
rt220_2          TRAY_220  at  ROLLER_TABLE_2
rt221_1          TRAY_221  at  ROLLER_TABLE_1
rt221_2          TRAY_221  at  ROLLER_TABLE_2.
```

The operator may enter any one as appropriate.

In the case of a **machine_lot** command, the prompt will be:

```
Enter PLAN_ID ?
Enter Plan Version Number ?
```

The following PLANs and their version numbers are supported now:

PLAN_ID	vsn	Part Type	NC used
LOK	1	Locking Clevis	lok1nc, lok2nc
LOK	2	Locking Clevis	lok3nc, lok4nc
LOK	9	Locking Clevis	robot moving only
CLA	1	Pipe Clamp	cl1nc (first cut)
CLA	9	Pipe Clamp	robot moving only
CLB	1	Pipe Clamp	cl2nc, (final cut)
CLB	9	Pipe Clamp	robot moving only
FLAM	1	Fluid Amplifier	flamnc
FLAM	9	Fluid Amplifier	moving part only.

The operator may enter any one as appropriate.

In the case of a **ship-tray** command, the prompt will be:

Enter SHIP_TRAY cmd file ?

The following ship tray cmd files are supported now:

st200_1	TRAY_200	at	ROLLER_TABLE_1
st200_2	TRAY_200	at	ROLLER_TABLE_2
st220_1	TRAY_220	at	ROLLER_TABLE_1
st220_2	TRAY_220	at	ROLLER_TABLE_2
st221_1	TRAY_221	at	ROLLER_TABLE_1
st221_2	TRAY_221	at	ROLLER_TABLE_2.

The operator may enter any one as appropriate.

3.2. Integrated Mode

The following steps are the instructions to set up and operate the VWS in an integrated mode.

(1). Power up all the VWS equipment
(2). Turn the Sun computer on
(3). Login as user: vws (use jun temporarily)
(4). Enter vc (for vws control)
(5). The VWS goes through the station cold boot procedure, sets up the windows, and displays instructions for further operator control.
(6). make sure the CMM system is up
(7). Type rcm (remote control mode) after seeing the operation mode selection message. The VWS is to be controlled by the AMRF Cell control system.

The system is now waiting for a SYNC from the Cell. A console message "waiting for SYNC" is printed every cycle until a SYNC is received. Once the SYNC message is received, the connection to the Cell is established, and the VWS is now under Cell control.

The system may use either the local database or the IMDAS. If the IMDAS is to be used, the user types rdm (remote data mode). The user may switch back to the local database system by entering ldm (local data mode). Refer to the Stand-alone Mode's Section (3.1) for explanation of these two commands.

The status window will display the following VWS status:

System Transition State:
COLD_SHUTDOWN,
WARM_SHUTDOWN,
WARM_STARTUP,
READY, or

```
        BUSY;

System Command Mode (cmode):
        LOCAL,
        REMOTE;

Database Mode (dmode):
        LOCAL,
        REMOTE.
```

3.3. Test and Maintenance Mode

One can set up the VWS control system in Test and Maintenance mode by typing a t in the console keyboard, anytime whether the system is in the integrated mode or in the stand-alone mode.

The VWS switches automatically into this "Test and Maintenance mode", when the system encounters a problem that may require operator intervention.

In this mode, the control is totally menu driven. With three levels of menus, an operator can select any type of action to take. An action here may be a simple "close vise", or may be a complicated task like "USE_NC testnc" (eg. use testnc to cut part), or "ROBOT_CHANGE_FINGER". On some actions, the operator will be prompted for further input. For example, when the operator chooses the action to rotate a robot gripper, the system will prompt for the amount of rotation. Lists of menus are shown in APPENDIX-1.

During the system testing, it is desirable to be able to run the system without using the equipment to move and cut the part. The VWS control system has this simulation capability - called simulation mode. The VWS may be brought up in simulation mode by the operator from the start. A capability to switch between real equipment control mode or simulated equipment control mode is also provided under this Test and Maintenance mode.

IV. THE VWS FUNCTIONAL DESCRIPTIONS

In the following, we will discuss:

* What data and plans are needed for the VWS ?
* How are these data and plans prepared?
* How are these plans and data used?

1. PROCESS PLANS AND THEIR PREPARATION

These plans and data are:

* the Cell level process plan,
* the workstation level process plans,
* the equipment level process plans, and
* data related to tray, part, quantity, schedule, etc.

1.1. The Routing Slip

The Cell level process plan, also called a routing slip, consists of a list of work elements (tasks) in a certain order. It is prepared by the Process Planning System for the Cell to coordinate the functions of various workstations[Ref9]. For example, the Cell will instruct the MHS (Material Handling System) to deliver a tray to a workstation, and then instruct that workstation to produce a part according to some workstation level process plan. And when the workstation finishes the part cutting process, the MHS is to pick up the part and deliver it to some other workstation. An example of a routing slip is shown in Fig.4 below:

```
        ........
        DELIVER_TRAY   (MHS) to VWS roller table #1
*       RECEIVE_TRAY   (VWS) to receive the tray
*       MACHINE_LOT    (VWS) to produce part
*       SHIP_TRAY      (VWS) to release the tray with part
        PICKUP_TRAY    (MHS) to pickup the tray with part
        DELIVER_TRAY   (MHS) to deliver to another workstation
```

****
Fig.4 Essential Information in a Routing Slip
****

Those tasks relevant to the VWS are marked with "*". Refer to the Cell document for more details of the routing slip.[Ref4]

When the VWS runs in the stand-alone mode, the workstation is under operator control; there is no routing slip for the Cell. Instead, an operator at the Sun computer terminal controls the system.

1.2. The Operation Sheet

The Workstation level process plan is called an operation sheet. The operation sheet is a list of tasks, or work elements, to be carried out by a workstation to accomplish a job (i.e. MACHINE_LOT). Currently, twenty work elements are supported by the VWS. APPENDIX-4 shows the complete set of work elements.

The operation sheet is workstation specific. The planner who prepares the operation sheet has to have the knowledge of that particular workstation's environment. However, in the VWS implementation, the preparation of the operation sheet is simplified. The planner only has to specify a few key points in a parametric form. No workstation internal structure is needed. The operation sheet is prepared by the Process Planning System or by the VWS operator using a text editor. Refer to Process Planning System documents for more details.

Figure 5 is an example of an operation sheet.

```
ROBOT_LOAD_PART
     WORK_PLAN          LOAD12          (*)
     PART               PART#12
     TO                 VISE

USE_NC
     DESIGN_FILENAME    demo1.pd        (**)

ROBOT_FLIP_PART
     WORK_PLAN          FLIP_PART_90    (*)
     PART               PART#12
     ROTATION_ANGLE     90

USE_NC
     PPLAN_FILENAME     demo2.pp        (***)

ROBOT_FLIP_PART
     WORK_PLAN          FLIP_PART_90    (*)
     PART               PART#12
     ROTATION_ANGLE     90

USE_NC
     MTOOL_FILENAME     demonc          (****)

ROBOT_UNLOAD_PART
     WORK_PLAN          ""              (*)
     PART               PART#12
```

****
Fig.5 An Operation Sheet
****

(*) Optional, if the WORK_PLAN name is not specified, the plan will be generated dynamically

(**) Part design document demo1.pd is used in this case, a process plan demo1.pp will be generated first, and then the NC code demo1.nc is generated. This statement is equivalent to "MTOOL_FILENAME demo1.nc".

(***) Process plan demo2.pp is used to generate demo2.nc. This statement is equivalent to "MTOOL_FILENAME demo2.nc".

(****) Process plan demonc is already in NC code.

1.3. The Instruction Set

The equipment level process plan specifies how each piece of equipment should act to accomplish a purpose. In AMRF terminology, it is an instruction set. However, the term "instruction set" is also specifically used in the VWS for the plan that the machine tool uses to cut the part. A plan that involves other equipment (i.e. robot, fixturing system, vacuum, roller table) is called an "equipment program" in this document. In the VWS, the equipment programs are not prepared by the process planning system. They are generated automatically by the Equipment Program Generator (EPG). More discussion on the EPG is given later.

In the VWS implementation, the instruction set (equipment level process plan as defined in the previous paragraph) may be generated automatically from the part design document. The NC code is generated automatically from the design document and the instruction set. For this reason, the NC file name identified in the operation sheet may be in three forms: NC, process plan, or the design document. No matter which form is given, it will be converted to NC code.

1.3.1. Numeric Control Code (NC)

If an instruction set is already in NC code format, it is readily usable by the GE-2000 milling machine controller to control the vertical machining center to cut the part.

In the VWS implementation, NC code is generated automatically by the system from a process plan and its associated design document. The instruction set exists in this form if the NC had been generated previously. A text editor may be used to edit the NC code to improve machine tool cutting efficiency. Figure 6 is an example of such an instruction set. This NC code is to be used by the GE-2000 controller. It is conformed to the EIA RS274D standard. See the GE-2000 programming manual in [Ref22] for details.

```
n0001 (ID,PROG,datxnc,data execution demo design,1)
n0002 g53
n0003 p69 = +0.735
n0004 p68 = +0.0
n0005 g90 g0 w(p69+(p68-10.5)) m6
n0006 p91 = 1.5
n0007 p12 = 91 m950
n0008 p90=50 p88=-.25 p89=40
n0009 p83=+16.825 p84=+7.425 p85=1
n0010 p70=0
n0011 g53 m9
n0012 g0 g90 m5 m6
n0013 g90 g0 x+36.5 y+15.0
n0014 ! Changing tool to probe for setting x_zero,y_zero
n0015 t(p89) m28 m67 m6
n0016 x(p83) y(p84)
n0017 (GSUB,OUTVWS)
n0018 p66=(p97+0.0) p67=(p98+0.0)
n0019 g56 g90 x(p66) y(p67)
n0020 ! 0.3 by 3 by 2.0 pocket
n0021 ! Changing tool to 0.625 inch diameter end_mill
n0022 g90 g0 m6 m9
n0023 g53
n0024 g90 g0 x+36.5 y+15.0
n0025 g90 g0 s2750 t12 d12 m3 m6
n0026 g56 g90 x(p66) y(p67)
n0027 m8
n0028 x+4.0 y+1.5
n0029 g0 z+0.1
n0030 g1 z+0.0 f5
n0031 x+3.0 y+1.5 z-0.2679
n0032 x+4.0 y+1.5 z-0.3
n0033 x+3.0 y+1.5
        .
        .
        .
        .
n0128 g53 m9 m5
n0129 g90 g0 w-9.0 m6
n0130 p91 = 0.0
n0131 p12 = 91 m950
n0132 g90 g0 x+0.5 y+19.5
n0133 (END,PROG)
```

****
Fig.6 An example of an NC program
****

1.3.2. Process Plan

The instruction set, such as "datex_plan_std.pp" in Figure 7, can be prepared by the Process Planning System [Ref12]. Alternatively, the process plan may be generated automatically by using the Automated Process Planner. Using the feature-based part design data as the source, the Automated Process Planner generates the process plan [Ref12]. In both cases, the AMRF standard process plan format is used. This type of instruction set has the file name extension ".pp", it is not immediately usable by the machine tool. The "NC code Generator" must be used to generate the NC code. The time it takes to generate an NC file from a process plan is less than a minute for a simple design.

```
--PROCESS_PLAN--

--HEADER_SECTION--
   PLAN_ID           := DATEX_PLAN;
   PLAN_VERSION      := 1;
   PLAN_TYPE         := INSTRUCTION_SET;
   DESIGN_ID         := DATEX_DESIGN;
   MATERIAL          := ALUMINUM;
   PROCESS_ENGINEER  := "VWS2 AUTOMATIC PROCESS PLANNER";
--END_HEADER_SECTION--

--PARAMETERS_SECTION--
   $$WORKPIECE       : WORKPIECE;
   $$TOOL_SET        : TOOL_SET;
        .
--END_PARAMETERS_SECTION--

--REQUIREMENTS_SECTION--
<<1>>  WORKPIECE
   ( WORKPIECE_ID    => $$WORKPIECE);
<<2>>  TOOL_SET
   ( TOOL_SET_ID     => $$TOOL_SET,
     COMPONENTS      => (3, 4, 5, 6, 7, 8));
<<3>>  TOOL
        .
--END_REQUIREMENTS_SECTION--

--PROCEDURE_SECTION--

<<1>>  INITIALIZE_PLAN
   ( PROG_NAME       => "DATA EXECUTION DEMO DESIGN",
     TIME            => "0000:01:00:00" );
<<2>>  SET0_CORNER
   ( TOOL_TYPE_ID    => PROBE_0.25,
     CORNER          => 1,
     X_OFFSET        => 0.0,
     Y_OFFSET        => 0.0,
     NEAR_X          => 16.825,
     NEAR_Y          => 7.425000000000001,
     PREC_STEPS      => (1),
     TIME            => "0000:01:00:00" );
<<3>>  MILL_POCKET
        .
--END_PROCEDURE_SECTION--

--END_PROCESS_PLAN--
```

****
Fig.7 An Example of Process Plan: datex_plan_std.pp
****

1.3.3. Feature-Based Part Design Data

A user with a design concept may use the Part Design Editor to design a part [Refll]. The output of this system is a feature based design geometry data description. The part design document has a file name extension ".pd". For example, a part design created during a demonstration may be named "demo.pd".

If the work element "USE_NC" in the operation sheet (Fig.5) has no value for the parameters "MTOOL_FILENAME" and "PPLAN_FILENAME", but does have a value for the parameter "DESIGN_FILE" (say demo.pd), the VWS will call the "Automated Processor Planner" to generate a process plan (demo.pp) and the "NC code Generator" to generate the NC (demonc). Figure 8 is an example of a design document "datex_design.pd".

```
(setplist 'datex_design '
(features    (features
               1   (1
                      feature_type              hole
                      center_x                  1
                      center_y                  1
                      diameter                  0.316
                      depth                     0.6
                      bottom_type               conical
                      countersink_diameter      0.5)
               2   (2
                      feature_type              pocket_corners
                      upper_l_x                 2
                      upper_l_y                 2.5
                      lower_r_x                 5
                      lower_r_y                 0.5
                      depth                     0.3
                      corner_radius             0.4
                      chamfer_in_depth          0.06)
               3   (3
                      feature_type              text
                      text                       "d"
                      font                      broad
                      lower_l_x                 3
                      lower_l_y                 1
                      height                    1
                      depth                     0.02
                      line_width                0.1356466
                      reference_feature         2))

 header      (header
               design_id   datex_design
               material    aluminum
               block_size  (block_size
                             length      6.95
                             width       2.975
                             height      0.735)
               description "data execution demo design")))
```

****
Fig.8 An Example of a Part Design Data
****

1.4. OTHER DATA AND PLANS

Other data and plans such as:

tray to be used (tray serial number),
tray type,
part blank to be used (part serial number),
tray location,
part description etc.,

are stored in the IMDAS and in the VWS' internal database.

2. HOW THE PLANS AND DATA ARE USED

The VWS has two modes of operation: the stand-alone mode and the integrated mode. In the integrated mode, the VWS is a component of the AMRF. It is directly under the control of the Cell. All interfaces with its peer workstations (MHS, TWS, HWS, IWS, CDWS) are done indirectly through the Cell.

In the stand-alone mode, an operator controls the VWS from the Sun computer console. The routing slip is not needed.

In the following, the example plans shown in the previous section are used to explain the VWS function. Please notice the granularity of each level of the process plan. The information in each plan is generic.

2.1. The Routing Slip

The Cell coordinates the activities of various workstations with a routing slip. An example of the information carried by a routing slip is shown in Figure 4. The meaning of the information in the figure is as follows:

step (1) - instructs the MHS to deliver a tray to one of the two roller tables.

step (2) - instructs the VWS to receive the tray from that roller table.

step (3) - tells the VWS to produce a part

step (4) - tells the VWS to release the tray with the produced part in it

step (5) - instructs the MHS to pick up the tray

step (6) - tells the MHS to deliver the tray elsewhere

2.2. The Operation Sheet

In the integrated mode, the name of the operation sheet comes with the Cell command "MACHINE_LOT". In the stand-alone mode, the operator is prompted for an operation sheet filename after the operator issues the command "m" (for MACHINE_LOT).

Figure 5 shows an example of an operation sheet. In the example, there are seven work elements for this job. The following is a step by step account of how the VWS uses the plan:

2.2.1. The First Step:

```
ROBOT_LOAD_PART
    WORK_PLAN    LOAD12
    PART         PART#12
    TO           VISE
```

This step is used for the VWS to move a part into the fixture. There are three attribute-value pairs for this statement. If the WORK_PLAN's value is specified (in this case, LOAD12), then this device program "LOAD12" is used to control the robot, vise, gripper, etc. to pick up the part and put it in the vise. In this case, the device program LOAD12 is pregenerated and stored as a file. LOAD12 is loaded when needed. This is the normal mode of operation.

If no program is specified, or the specified program is non-existent, the rest of the attribute-value pairs are used to generate the device program dynamically for the task. The program generation is done by the EPG. Refer to the EPG module description for more details regarding the program generating process. However, this is riskier. Equipment could be damaged if any error in the device program because of bugs in the program generation logic.

2.2.2. The Second Step

```
USE_NC
   DESIGN_FILENAME      demo1.pd
```

means: use the design document "demo1.pd" to cut the part in the fixture.

The work element USE_NC has a single attribute_value pair. In this case, it is "DESIGN_FILENAME demo1.pd". It tells the VWS to use the design document "demo1.pd" to cut the part. As explained before, the instruction set may come in three forms. They are characterized by the attribute "DESIGN_FILENAME" and the file name extension. The file name extension ".pd" of demo1.pd indicates that the instruction set "demo1" is in the feature-based design format, and was generated by the VWS Part Design System.

After the VWS determines that the incoming file is in this format, the Process Planner is called to process "demo1.pd" and to generate a file "demo1.pp" in the AMRF standard process plan format. This process plan "demo1.pp" and the part design data "demo1.pd" are used by the NC code Generator to generate an NC program "demo1.nc".

Once an NC program is ready, it will be downloaded to the GE-2000, the vertical machining center controller. The GE-2000 controller has enough room to store dozens of NC programs.

To start the vertical machining center requires a two-step procedure. First the correct NC program is selected by the GE-2000 controller, and then a "CYCLE_START" command is given to the controller to start the machine.

To summarize, the single work element "USE_NC" is broken down by the VWS automatically into several tasks:

(1). GENERATE_NC
From demo1.pd -> demo1.pp -> demo1.nc

(2). DOWNLOAD_NC_HP
To download "demo1.nc" from Sun to HP-9000

(3). DOWNLOAD_NC_GE
To download "demo1.nc" from HP to GE-2000

(4). SELECT_NC name
GE-2000 to select the "demo1.nc"

(5). CYCLE_START
To tell the controller to start to execute the program selected

These workstation dependent tasks are generated by the VWS and

carried out one by one to accomplish a goal - to "USE_NC demo1.pd" to cut the part.

2.2.3. The Third Step:

```
ROBOT_FLIP_PART
    WORK_PLAN           flip_part_1
    PART                PART#12
    ROTATION_ANGLE      90
```

This work element tells the VWS to flip the part to the other side. This work element is similar to the first one (ROBOT_LOAD_PART). In this case, a robot program "flip_part_1" is specified. This robot program will be used to flip the part by 90 degrees so that the next side can be cut.

To flip the part, many steps are involved. First, the robot has to get ownership of the machine tool table. Then the vise is opened, the robot with proper gripper picks up the part, and the gripper rotates the part 90 degrees. The robot then puts the part back into the vise. Once the part is in the vise, the vise is closed to secure the part, and the robot is moved to its home position. Then the robot's ownership of the machine tool table is released. All these steps are specified in the equipment program - flip_part_1.

As in step 1, the other two parameter pairs in this example are not used. Had the value of WORK_PLAN been nil, the EPG would have been called, using the other parameters, to generate the robot flip part program for this occasion.

2.2.4. The Fourth Step

```
USE_NC
    PPLAN_FILENAME      demo2.pp
```

means: "use the process plan demo2.pp to cut the part".

The instruction set "demo2.pp" is used to cut the next side of the part - because the part has just been flipped 90 degrees by the previous step. This step is similar to the second step, the only difference is the use of "demo2.pp" instead of "demo1.pd". Notice the instruction set "demo2.pp" has an extension ".pp". It means the plan is in the AMRF standard process plan format. "demo2.pp" will be processed to generate the NC, "demo2.nc", which is then used by the machine tool to cut the part. Refer to step 2 for more details.

2.2.5. The Fifth Step:

```
ROBOT_FLIP_PART
    WORK_PLAN           flip_part_2
    PART                PART#12
    ROTATION_ANGLE      90
```

Similar to the third step except the equipment program flip_part_2 is used to turn the part.

2.2.6. The Sixth Step

```
USE_NC
    MTOOL_FILENAME      demonc
```

means: "use the NC program demonc to cut the part".

The NC program "demonc" is used to cut the next side of the part. This step is similar to the second step, the only difference is the instruction set is already in NC format, ready to be downloaded to the GE-2000 controller. See step 2 or 4 for details.

2.2.7. The Seventh Step (last step),

```
ROBOT_UNLOAD_PART
    WORK_PLAN   ""
    PART        PART#12
```

means: "move part#12 from the vise to the tray area".

The robot is back in action again. But this time the robot transfers the finished part back to the tray. The parameters have the same meaning as the first step, except this one does not have the "TO" destination attribute_value. This is because the implementation assumes that the finished part is destined to go to the same location on the same tray it came from. If the WORK_PLAN's value (file name) is specified and an equipment program file of that name exists, then the equipment program is used to control the robot to perform the unloading task.

If NO equipment program is specified as the value of WORK_PLAN, as in this case, or the specified plan is a non-existent file, the Equipment Program Generator is called to generate one. The parameter indicates that a part named "PART#12" is to be unloaded. The location of "PART#12" is retrieved from the VWS

internal world model. The Equipment Program Generator (EPG) uses this information to generate an equipment command procedure that includes steps like open the vise, lock and unlock the vertical machining center table, etc.

2.3. How The Instruction Set Is Used

The machine level process plan (the instruction set) is a set of instructions to tell the machine tool how to cut the part. The VWS gets the instruction set through the parameters in the operation sheet.

Figure 5 shows the way an instruction set is specified by an operation sheet.

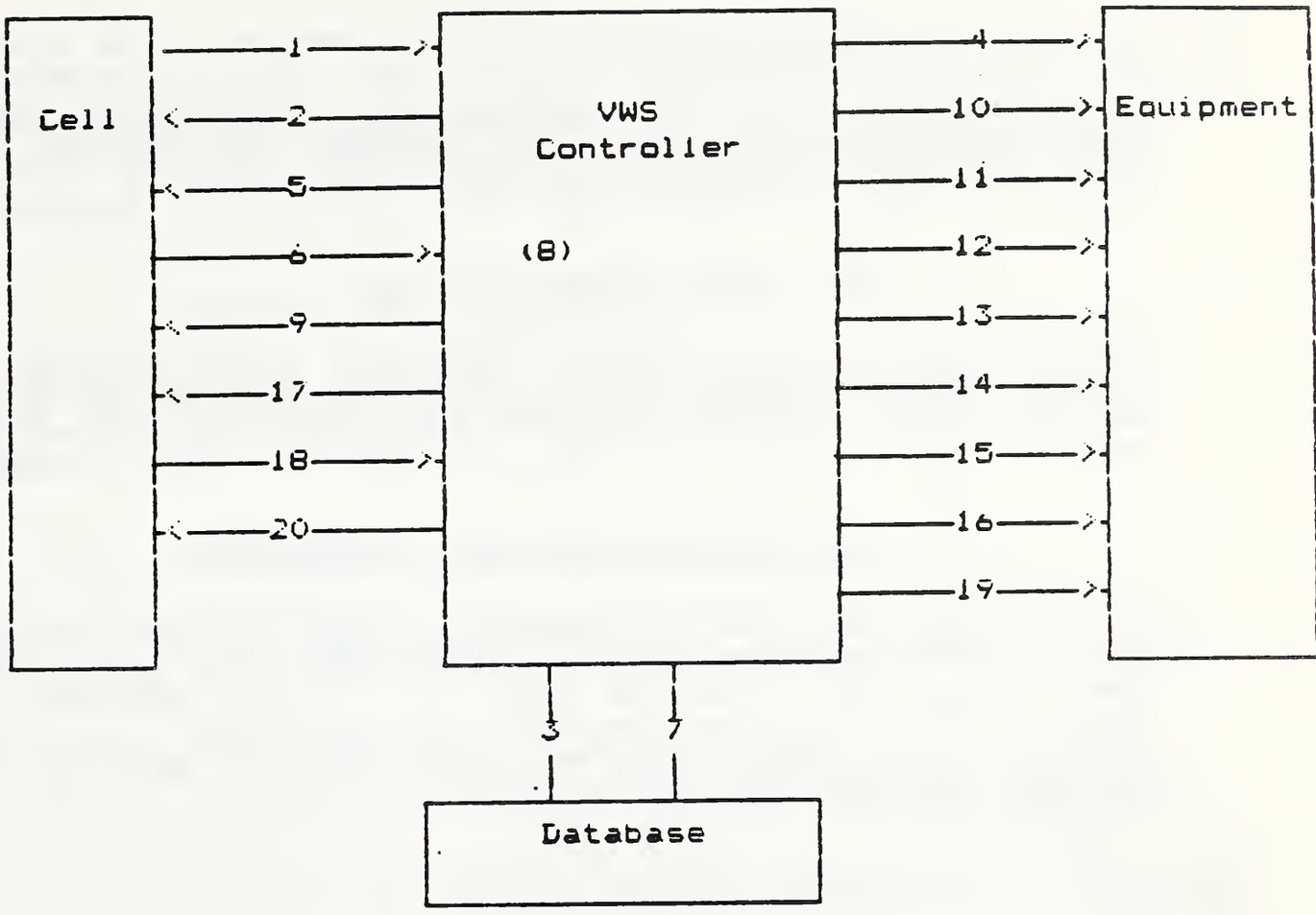
Refer to section 2.2 for details.

V. THE VWS CONTROL SEQUENCE DESCRIPTION.

Once the necessary process plans and data in the database are prepared, the VWS is ready to cut the part. The following scenario is a summary of VWS functionality represented in a time sequence - with the following assumptions:

* The VWS runs in the AMRF integrated mode under Cell control.
* PART#12 in the tray is the part blank.
* Part design document "demo1.pd", the process plan "demo2.pp", and NC file "demonc" have been prepared.
* The operation sheet shown in Figure 5 is used.

Figure 9 shows the sequence of actions. The numbers indicate the sequence of events, and the arrow indicates the information flow. A brief discussion of each event is given below.



1. RECEIVE "RECEIVE_TRAY" COMMAND FROM CELL
2. SEND "BUSY" STATUS TO CELL
3. RETRIEVE TRAY AND PART DESCRIPTION
4. LOCK THE ROLLER TABLE
5. SEND "DONE" STATUS TO CELL TO INDICATE TRAY IS RECEIVED
6. RECEIVE MACHINE_LOT COMMAND FROM CELL
7. RETRIEVE THE OPERATION SHEET
8. PRE-PROCESS THE OPERATION SHEET
9. SEND "BUSY" OR "ERROR" STATUS TO THE CELL
10. ROBOT LOAD PART
11. DOWNLOAD NC (demo1.nc)
12. MACHINE THE PART
13. ROBOT FLIP THE PARTIALLY CUT PART
14. DOWNLOAD NC (demo2.nc)
15. MACHINE THE PART (cut the other side)
16. ROBOT UNLOAD PART
17. SEND STATUS TO CELL
18. RECEIVE SHIP_TRAY COMMAND FROM CELL
19. RELEASE TRAY
20. SEND STATUS TO CELL

Fig.9 Sequence of Actions

1. RECEIVE_TRAY

After confirming that the tray has been delivered to the VWS's part receiving roller table by the robot cart, the Cell sends a "RECEIVE_TRAY" command to the VWS.

2. SEND "BUSY" STATUS TO CELL

The VWS sends a busy status to the Cell to acknowledge that the RECEIVE_TRAY message has been received, and the VWS is actively working on it. The VWS decodes the command to find out which roller table receives the tray and the tray serial number.

3. RETRIEVE TRAY AND PART DESCRIPTION

Based on the tray serial number, the VWS retrieves the tray type, the tray contents, the part serial number, the part location and description, etc. from the IMDAS or the local database. The VWS keeps a set of information about its environment called the VWS internal world model. The VWS internal world model is updated to reflect the part and tray data.

4. LOCK THE ROLLER TABLE

The VWS will claim ownership of the tray that is received by physically locking the roller table. The hardware locking mechanism is to prevent the tray from being taken away by the robot cart. This hardware locking procedure is important for the safe operation of the robot.

5. SEND "DONE" STATUS TO CELL

The VWS sends a "DONE" status to the Cell indicating that the tray has been properly received, and its associated data has been retrieved from the database. An "ERROR" status will be sent if any error occurred.

6. RECEIVE MACHINE_LOT COMMAND FROM CELL[2]

---

[2]Since there are two roller tables, up to two RECEIVE_TRAY commands could be received before the VWS gets a MACHINE_LOT command.

The Cell will now send a MACHINE_LOT command instructing the VWS to produce the part. This command comes with information of PLAN_ID and PART_ID. It instructs the VWS to use an operation sheet specified by the PLAN_ID to cut a part specified by PART_ID. Database activities are needed to find this information.

7. RETRIEVE THE OPERATION SHEET

The operation sheet specified in the PLAN_ID field of the MACHINE_LOT command is retrieved from the database (local or global).

8. PREPROCESS THE OPERATION SHEET

The VWS scans through the operation sheet. Some data or plans called for will be retrieved. In this example, there are five tasks involved.

Task#1 and task#3, involving robot part transfer actions, are specified by the device programs "LOAD12" and "FLIP_PART_180". These two programs could have been pregenerated, and these programs will be retrieved from the database. On the other hand, there is no program specified in task#5. This task is to unload the finished product from the fixture back to the tray. The part unloading program will be generated later.

In task#2 and task#4, the part design file name "demo1.pd" and the process plan "demo2.pp" are retrieved, and converted into the NC form, demo1.nc and demo2.nc, respectively.

9. SEND "BUSY" OR "ERROR" STATUS TO THE CELL

The VWS will send a "BUSY" status to the Cell if all the preprocess activities of the operation sheet are successful. That is:

* All pregenerated robot programs are retrieved successfully.
* All instruction sets specified exist or were generated.

If any one of the above is negative, an "ERROR" status will be sent instead.

10. ROBOT LOAD PART

The VWS is ready to start producing the part. The first thing to do is to move the part to the fixturing system. The equipment program "LOAD12" to transfer part#12 from the tray to the vise is checked for its existence. If the program exists, it is used. Otherwise, program LOAD12 will be generated dynamically.

11. DOWNLOAD NC (demo1.nc)

The VWS downloads NC "demo1.nc" to the vertical machining center.

12. MACHINE THE PART

The VWS instructs the GE-2000 to select the NC program to be used (in this case, demo1.nc) and then execute the program with a "CYCLE_START" command.

13. ROBOT FLIP THE PARTIALLY CUT PART

The equipment program "FLIP_PART_1" is used to turn the part to the other side. If the program exists, it is used. Otherwise, program "FLIP_PART_1" will be generated dynamically. Refer to 2.2.3.

14. DOWNLOAD NC (demo2.nc)

The instruction set "demo2.pp" has been converted into NC form "demo2.nc". This NC file is then downloaded to the machine tool.

15. MACHINE THE PART (cut the other side)

The VWS instructs the GE-2000 to select the NC program to be used (in this case, demo2.nc), and then executes the program with a "CYCLE_START" command.

16. ROBOT UNLOAD PART

Since the robot program is not provided in this example, the VWS calls the EPG to generate the equipment control program to transfer the produced part back to tray on the roller table. This completes the machining process. The internal database is updated to reflect the current workstation status.

17. SEND STATUS TO CELL

Job "DONE" status is sent to the Cell. If the job is incomplete for any reason, an "ERROR" status will be sent instead.

18. RECEIVE SHIP_TRAY COMMAND FROM CELL

When the Cell receives a "DONE" status from the VWS, it will ask the VWS to release the tray with the finished part in it by sending the "SHIP_TRAY" command.

19. RELEASE TRAY

In response to SHIP_TRAY, the VWS updates the IMDAS and its VWS internal world model, and physically unlocks the roller table which has tray#12 on it. By doing this, the VWS no longer has ownership of the roller table. The robot cart which is controlled by the MHS may come in any time to pick it up and deliver it elsewhere. (The update function of the IMDAS has not implemented yet.)

20. SEND STATUS TO CELL

A "DONE" status is sent to signal the Cell that the VWS has completed the SHIP_TRAY task.

VI. THE VWS SOFTWARE PACKAGES

Software packages important to the VWS are described below. These are classified into five general areas: control, utility and maintenance, process planning, communication, and database. Some descriptions refer to other documents when appropriate.

1. CONTROL RELATED SOFTWARE

In this section, the software related to the control function of the Vertical Workstation system is discussed. The core of the VWS control systems is the PMOS, the Process Management Operating System.

The PMOS controls a set of finite state machine modules (FSMs). Each FSM has a group of supporting functions. Other functions like preparing the process plan, automated NC generation, preparing an FSM, etc. are considered as utilities and are discussed under other software packages.

PMOS

The PMOS is the backbone of the control system. It schedules, monitors and executes a set of FSMs just like other operating systems execute a set of application programs. It provides a rich repertoire of commands for real-time control by an operator from the console keyboard. Taking advantage of the large screen and the powerful window system provided by the Sun computer, six windows are used to control and to display the system status. System status can be selectively displayed, logged to a file, or both in real-time.

The PMOS logically executes a set of FSM's in parallel in each cycle. This is different from most operating systems which run a set of application programs in a sequential manner. In the PMOS system, all inputs and outputs are done in a time-slice between cycles. This prevents race conditions from developing where modules may execute asynchronously.

But this technique creates an interesting side effect. Unlike the conventional operating systems, there are no i/o completion or maximum time-slot rules, or priorities to be used for the selection of the next module to execute. All modules are to be checked in each cycle. In a single processor computer, this creates overhead. A wake-up and sleep mechanism is implemented to alleviate this problem. Each FSM module may set itself into sleep for any period of time. Each module may wake-up any number of modules immediately (in the next cycle) or

certain cycles later on if there is a need. With this facility, the number of modules to run in each cycle is greatly reduced.

To improve the efficiency further, the PMOS rearranges FSM modules into a run-time format during the module loading process. This process improves the execution speed by a factor of 2.

Further improvement may be attained by compiling the LISP code. Since this is a research prototype and the speed is adequate so far, for ease of maintenance and development work, the LISP source code under the interpreter is used.

FSMs (Finite State Machine Modules) Currently a set of six finite state machine modules is used for production control. These are:

1) FMP - Front-end Message Processor
2) TM - Transition Manager
3) PRO_MGR - Production Manager
4) QUE_MGR - Queue Manager
5) DSP_MGR - Dispatch Manager
6) EQUIP_MGR - Equipment Manager

PRO_MGR, QUE_MGR, and DSP_MGR control the flow of the manufacturing process. They are called the Manufacturing Manager (MM). These six FSMs together with their supporting functions determine the basic sequencing of activities of the workstation and its equipment level systems. A FSM is defined by a LISP property list structure [Ref19]. Each module has the following structure:

1) variable declarations,
2) pre-process calls,
3) a state-table,
4) post-process calls, and
5) local functions.

The state-table portion of the module is further divided into a condition section and an action section. Each line in the table associates conditional tests that define the state with actions to take when in that state. This may be viewed as a collection of "if...then..." rules. In this respect, it is similar to the production rule-based system, except the rules are arranged in a tabular fashion. The average size of a table used in the VWS control system is about 22 by 9, (with 6 conditions,

3 actions, and 22 rules). For a state-table implementation, this is not small, but is still manageable.

A brief description of each module is in order:

1.1. Front-end Message Processor (FMP)

The FMP is a collection of modules handling the operator interface and the communication functions. The main module is "oper.1" and is controlled by a state-table module "oper.fsm". The FMP uses five windows for displaying the command menu, system statuses, and diagnostic messages, etc.

The FMP passively monitors the keyboard. Any operator input is interpreted and acted upon accordingly. A rich set of commands is provided to control the workstation. These commands fall into three catagories:

1.1.1. preemptive commands

!	refresh the display windows
b	break the program
gc	initiate the garbage collection process
t	set the VWS into "TEST & Maint. MODE"
f	set or reset system flags
qlcm	change to stand-alone mode - quick
qrcm	change to integrated mode - quick
rdm	use remote database (IMDAS)
ldm	use local database (Sun file system)

1.1.2. Stand-alone mode commands

x	cold shutdown
r	startup
s	warm shutdown
ldm	set to local database mode
rdm	set to remote database mode
rcm	set the VWS to run in integrated mode
a	attach a piece of equipment to the VWS
d	detach a piece of equipment from the VWS
rt	receive a tray
st	ship a tray
m	machine a part (machine_lot)

1.1.3. Integrated mode commands

COLD_SHUTDOWN	shutdown the VWS cold
WARM_STARTUP	startup the VWS

WARM_SHUTDOWN	shutdown the VWS, may be restarted later
SYNC	establish connection with the Cell
RESET	reset the VWS (to recover from error)
ATTACH	make connection with a piece of equipment
DETACH	detach connection from a piece of equipment
CHANGE_DMODE	change database mode (local or remote)
CHANGE_CMODE	change command mode (local or remote)
RECEIVE_TRAY	receive a tray
SHIP_TRAY	ship out a tray
MACHINE_LOT	VWS to machine a part

1.2. Transition Manager (TM)

A framework for controlling initialization, restart, shutdown, and reconfiguration of the AMRF systems was defined and presented by Dave O'Halloran[Ref14]. This framework was partially implemented by O'Halloran for the VWS. It is in the form of a state-table module under the control of PMOS.

The TM is responsible for the system startup, shutdown, and restart tasks. It also coordinates change of mode of operation from stand-alone to integrated mode, or vice versa. In the future, it will also manage aspects of error recovery.

1.3. Production Manager (PRO_MGR).

After receiving a job from FMP, the PRO_MGR retrieves the job description list (the operation sheet) and the corresponding instruction sets and equipment control procedures. In the VWS implementation, if the instruction set is already in the NC form, the PRO_MGR retrieves NCs from either the local or the IMDAS. If the instruction set is in the form of a feature-based part design document, or the AMRF standard process plan form, either the Auto-Process-Planner or the NC-Generator or both are called to generate NC for the vertical machining center [Ref13]. The equipment control procedures may be specified by some equipment program names, or they may come as work elements with parameters. If program names are used, these programs are retrieved. At any rate, the PRO_MGR makes sure that all the necessary information and the resources needed for the job order are complete. The PRO_MGR then decomposes the job into tasks according to the order in the operation sheet. These tasks are then passed to the next level, the QUE_MGR, one by one.

1.4. Task Queue Control Manager (QUE_MGR).

The queue manager keeps track of the flow of tasks to different pieces of equipment. There is only one queue manager in the VWS. In a more complicated system, one might want a separate queue manager for each piece of equipment.

1.5. Dispatch Manager (DSP_MGR).

It accepts a task from the QUE_MGR, dispatches the task to its piece of equipment, and monitors the progress of the task. When the task is completed, it will request the next task. Again, in the VWS, only one dispatcher is adequate for our purpose.

The Production Manager, the Queue Manager, and the Dispatch Manager form an effective means of taking a job (operation sheet), breaking it into tasks, and sequentially sending the tasks to the Equipment Manager. Together, these three are referred to as the Manufacturing Manager in other parts of this document.

1.6. Equipment Manager

The Equipment Manager consists of a group of programs: sub.fns.1, sub.fnsa.1, control.1 etc. The sub.fns.1 takes a job from the MM, and breaks the job down according to the job table in the sub.fnsa.1 program into tasks for the subordinate devices. The job in this level may be ROBOT_LOAD_PART, ROBOT_UNLOAD_PART, ROBOT_FLIP_PART, GENERATE_NC, DOWNLOAD_NC, USE_NC, etc. The list of tasks is called the device program in this document. The device program is then passed to the control.1 program, which then sends one task at a time to the intended device to carry it out.

Refer to APPENDIX-2 for a pseudo-code version of "control.1".

2. UTILITY AND MAINTENANCE-RELATED SOFTWARE

2.1. State-table Module Editor (STE)

This utility provides an efficient way of writing or editing state-table modules. The state-table module is in the form of a spread sheet. The writing of a state-table module involves a lot of column and row manipulations. The STE makes the job of preparing a state-table module much easier.

The STE has a rich set of instructions developed by us to manipulate columns and rows. Refer to APPENDIX-3 for a listing of the state-table editor command menu.

2.2. Automated Equipment Program Generator (EPG)

The EPG is a situation-driven equipment program generator. It generates programs, some of which include conditional statements, based on the situations currently stored in the internal database. The EPG features are as follows.

2.2.1. On-line generation

When processing the work element such as ROBOT_LOAD_PART, ROBOT_UNLOAD_PART, or ROBOT_FLIP_PART, with no WORK_PLAN specified, the EM will call the EPG to generate a program to control the equipment. Using the parameters of the work element and the VWS internal world model, the EPG creates an equipment program (command procedures) to coordinate the robot, the vise, one of the roller tables, and the gripper to accomplish the work required for that particular situation. This feature is particularly helpful when a variety of parts must be handled and each part potentially requires a unique non-reuseable program.

2.2.2. Off-line generation

The EPG can generate an equipment program off-line. This off-line capability is helpful for device program testing and development. Tested programs provide safety assurance.

In off-line use, the Generator also uses current data in the database. This implies that the database must simulate the beginning environment of a task. Because manual data setting is tedious, a menu-driven tool was developed to help the process.

2.2.3. Situation-driven

The EPG generates different programs depending on the run-time situation. As an example, consider part handling that requires setting up the robot gripper prior to holding the part. A program that includes finger attachment would be generated when the data indicates that a part to be manipulated requires the fingers and that the fingers are not currently attached to the robot.

2.2.4. Conditional-statements

A program can include conditional statements to handle the real-time situation. For example, prior to picking up a part, the

type of finger attached to the end-effector is checked for the correct type. If a wrong type of finger is attached, correction action will be taken. This is an equivalent to an "IF...THEN..." type of statement.

2.2.5. Program_construct_and_features.

Programs are constructed with high-level programming language structures such as: conditional statements, labels, subroutine calls, and branching. A program structure and its collection of features are presented in APPENDIX-5.

2.2.6. Supported tasks.

The EPG currently supports transferring parts both with or without a flip. Examples of the supported part transferring tasks are ROBOT_LOAD_PART and ROBOT_UNLOAD_PART. Both tasks has parameters. In the case of ROBOT_LOAD_PART, the parameters are the part id (say WIDGET) and the destination (say VISE) for the part. From the part id, WIDGET, the EPG obtains its location and its size. From the destination parameter VISE, the EPG find the current status of VISE from the database. The EPG uses these information to generate the device programs.

In addition to the basic operation generated in a program as described in the ROBOT_LOAD_PART example, the EPG also handles the operation of flipping a part a certain amount. As an example, consider that another widget would be machined. But this time, the widget requires machining on two opposite sides. After the first side has been machined, the widget would be flipped 180 degrees so that its other side can be machined. For this requirement, the EPG generates a program for:

* fetching the widget from the vise,
* flipping it 180 degrees,
* placing it in the vise, and
* locking the vise.

A task for vacuuming is currently under development. Together, these tasks cover what the VWS requires for many machining application.

Final notes: The concepts for the EPG are an extension of what has been referred to as the "blocks world" concepts found in a LISP book by Horn and Winston[Ref19]. For example, to pick up a block in the blocks world when the block currently supports another block, this other block must first be cleared. Similarly, to place a part in a vise when the vise is closed, the vise must be opened first.

The automated generation of a program uses a straightforward procedural approach. It is not rule based. No attempt has been made to optimize the robot path in transferring a part from one location to another. This approach is satisfactory for the VWS because robot motion occupies only a minor portion of the operation time.

Refer to [Ref6] for more description.

2.3. Equipment Command Verifier: ECV

The ECV is used at the VWS for verifying whether it is safe to execute an equipment command. For example, to have the robot place a part in the vise while the vise is closed would not be allowed.

Before sending a command to the equipment interface processor, the EM will call the ECV to examine the command. Any command judged unsafe by the ECV will NOT be executed.

The ECV verifies one command at a time, it DOES NOT keep history. Using the data in the internal data model at that moment and a set of safety rules, the ECV analyzes the command for that circumstance.

The ECV is a rule-based system with three major components: an inference module, a rule module, and an environment data model. Each command to be verified enters this system and causes only a subset of rules and data corresponding to the command to be used. The rules come from a super-set of all rules relevant to the workstation contained in the rule module. Similarly, the data come from the database that reflects the current knowledge of the robot world.

Through forward chaining, each of the rules is checked against the data. The checking is done by matching the conditions portion of the rules to the data. If there is any match, the matched rule is remembered. Whether a command passes the verification test depends on the remembered rule(s).

The inference module prepares and activates a set of rules according to the data for a particular type of command that will be verified. For example, a robot-move command would call for all rules and current data related to the robot-move: rules and data for speed, data for where the robot is, and so on. Since only a limited set of corresponding rules and data are used, the processing time for checking is shorter; thereby potentially more efficient for a large system.

Each rule consists of a set of conditions and a set of actions. The conditions and actions can be expressed as patterns if desired. An example rule for a command type "move" might be: if robot whereabouts are unknown, then don't move. Another rule might be: if the command is to "approach" an object and if current robot speed is above allowable approach speed, then don't "approach".

Currently the ECV is a small system - with fewer than 20 rules. It shows that, in principle, verification can be done in this way. By adding more rules, it can become a comprehensive verification system.

Refer to [Ref6] for more details.

2.4. Test and Maintenance Program

This set of utilities was originally developed for an operator to control and test various equipment components on a step by step basis. It has evolved into a very useful package, a full feature off-line control system. Besides being able to interrogate each device on an atomic instruction level, it may also control several devices together to perform a task under a device control program. For example, one may use it to find the location of the robot (WHERE_ROBOT), to close or open a vise, to lock a roller table, or to perform a task (specified by a device control program) like transferring a part, which involves locking the roller table, locking the machine table, opening the vise, controlling the robot to pick up the part from the tray and move it into the vise, closing the vise, unlocking the machine table, updating the internal database, and possibly the external database, etc.

The Test and Maintenance Program is completely menu driven. An operator can control the VWS effectively with very few key strokes. This package may be initiated from the console at any time. It is automatically triggered by the VWS control system when an error or unexpected condition occurs that needs operator intervention. In the future, we will try to expand this package into the AI domain, where the operator will be replaced by an "EXPERT" system.

Refer to APPENDIX-1 for a listing of the command menu.

2.5. Trace System

An extensive trace and diagnostic capability is built into the system. Various levels of system information may be displayed on

the screen or logged onto files by a command by the operator. The VWS has the following tracing capabilities:

* State-table module traces,
* Device control sequences tracing the command and status dialogue,
* Events tracing.

3. PROCESS PLANNING RELATED SOFTWARE

One of the implementation goals is for the VWS to be able to run as a self-sufficient stand-alone workstation. A significant portion of the software is devoted to the process planning area. The VWS has demonstrated the capability of being able to accept an equipment level process plan created by the AMRF Process Planning System. But the majority of the equipment level process plans come from the Automatic Process Planner using the part design document that is prepared by the Feature-based Part Design System. In the following, a short description for each system is given.

3.1. Feature-based Part Design System

The part design editor (PDE) is used to generate feature based part design document, so it is a CAD system for the stand-alone VWS. The system is menu driven, has graphic display and includes an extensive design verification system. Using this system, a part can be designed quickly and efficiently.[Ref11]

3.2. VWS Process Planner

The VWS Process Planner automatically generates a process plan from the part design document output from the PDE. Process plans generated this way conform to the AMRF standard text format. These files may be modified with any text editor if desired. [Ref12]

3.3. VWS NC code Generator

The NC code Generator is used to develop NC code for the GE-2000 controller that controls the vertical machining center. The input to the Generator is a process plan and its associated part design document. For most of the parts designed in the VWS, the NC code generation process can be done within two minutes.[Ref13]

4. COMMUNICATION RELATED SOFTWARE

4.1. AMRF Mailboxes

The VWS communicates with the outside world, including the Cell and the Imdas, through AMRF MAILBOXes [Ref15]. If the system is running in stand-alone mode and the local database is used, this subsystem is not activated. It is initiated by entering either

"rcm" (remote control mode)

or

"rdm" (remote database mode)

from the console. This will allow the VWS to establish connection with the Cell or the IMDAS. If for any reason, a stand-alone mode is desired, the command:

"lcm" (local control mode)

may be issued. The VWS will ignore the commands coming from the Cell. Similarly, the IMDAS may be turned off by issuing the command:

"ldm" (local database mode)

to ignore the communication channel. In both cases, the communication subsystem is not actually turned off. It is simply ignored in the present implementation of the VWS.

Currently, The established AMRF MAILBOXes are:

C_VWS_CMD - Cell CMD input mailbox

C_VWS_STS - for VWS status to the Cell

DS_VWS_CMD - for commands to the IMDAS

DS_VWS_STS - for status messages from the IMDAS

DS_VWS_DIN - for data received from the IMDAS

DS_VWS_DOUT - for data being sent to the IMDAS

4.2. Communication With The VWS Equipment

Communications between the workstation controller (Sun computer) and its equipment-level components is considered as an internal

matter in this implementation, and do not make use of the mailboxes described above.

The following connections use an RS-232 interface:

* VWS controller (Sun computer) with the CEC (HP-9000),

* The CEC with equipment components such as the robot controller, the roller table controller, the GE-2000 controller for the vertical machining center.

The messages from the VWS controller to the CEC (HP-9000) consist of commands and inquiries. The HP-9000 accepts the command message, sends an acknowledgement to the controller, and sends the message to the intended device controller to carry out the action. The HP-9000 does not voluntarily return status. All status reports are initiated by a status inquiry from the VWS controller.

An example of a vise open command sequence is in order:

VWS controller	CEC (HP-9000)
Controller sends Command with a sequence number to HP-9000 to open the vise: VISE-OPEN.	
	HP-9000 acknowledges the cmd received, and passes the command to the vise.
After receiving the acknowledgement, the controller issues a status query command: VISE_CMD STATUS.	
	HP-9000 Reads vise status. If the vise is still working on the command, the HP-9000 sends an equivalent of "UNDONE" status to the VWS controller. Actually. the message sent consists of a sequence number and "DONE" with an old sequence number.
When a status is received, the sequence number is checked to see if it matches the last command sent. If the sequence number fails to match, it is an "UNDONE" status. The controller waits a little bit, and reissues the status query: VISE_CMD STATUS.	
	Reads vise status again. If the vise has opened, then a "DONE" status is sent to the controller.
When a "DONE" status is received, it means the vise has acted upon the command. Whether the vise has successfully opened, and the size of the opening is still unknown. To find out, the controller has to issue another vise status query: STATUS VISE.	
	Upon receiving the STATUS VISE command, HP-9000 reads the vise status and sends it to the controller.

The controller receives the vise status. This completes the command and status information exchange for a simple command "VISE OPEN"

5. DATABASE RELATED SOFTWARE

Data such as the operation sheet, the feature-based design document, the process plan, and the NC's are stored in either the IMDAS or the VWS local database. The Operator may switch between these two databases at any time. The VWS database management system makes the database system being used transparent to the user.

Refer to [Ref8], the VWS database system document for more details.

APPENDIX-1 TEST and MAINTENANCE MANAGER Operator Command List

TOP LEVEL	2nd LEVEL	3rd LEVEL
COMMAND		
	ROBOT	
		WHERE_ROBOT
		SPEED ?
		DO MOVE SAFE
		DO MOVE WV?
		DISABLE_TRAY?
		RELEASE_TRAY?
		DISABLE_TABLE
		RELEASE_TABLE
		EXECUTE ?
	GRIPPER	
		OPEN
		CLOSE
		ROTATE_?
	HYDRAULIC	
		CLAMP_PALLET
		RELEASE_PALLET
		LOCKALL
		UNLOCKALL
		PRESSURIZE_HOLE
		DEPRESSURIZE_HOLE
	VISE	
		OPEN
		CLOSE
		DISABLE_TABLE
		RELEASE_TABLE
	VACUUM	
		ON
		OFF
	MONARCH	
		CYCLE_START
		RELEASE_TABLE
		STATUS
		DOWNLOAD_HP
		DOWNLOAD_GE
		TABLE (?)
		PROGRAM_SELECT(?)

PROGRAM_DELETE(?)

```
STATUS
                    GRIPPER
                    CHANGER_1
                    TABLE
                    VISE
                    TRAYS
                    VACUUM
                    TLI_TABLE
                    WHERE_ROBOT
                    PALLET
                    MONARCH
                    ROBOT_CMD
                    MONARCH_CMD
                    GRIPPER_CMD
                    VACUUM_CMD
                    VISE_CMD
                    HYDRAULIC_CMD

TEST_EQUIP
                    ALL
                    ROBOT
                    MONARCH
                    VACUUM
                    TABLE
                    TRAY1
                    TRAY2
                    VISE
                    HYDRAULIC

PEEK
DOWNLOAD_NC
EXECUTE_NC
BREAK
QUIT
```

APPENDIX-2 Device Program Executor (pseudo code of control.1)

```
control (plan)

  loop
        if end of plan, job done
        get a command from the plan
        call ECV to verify this command
        call operator intervention if problems
  re_send
        send this command to HP-9000
  read status
        if no status return (time_out) go re_send

  status_returned
        if the command was a STATUS request command, then
                process_status plan status
                (call operator intervention if error cond.)
        endif
        if command is the dev_CMD type, then
                process_dev_CMD (plan status dev_type)
        endif
  next
        (go loop)
```

APPENDIX-3 A List of State-Table Editor Commands

State-table EDITOR MENU

```
addvart <variable type etc> -- add a type to a variable.
at      <col> -- add tag and its def into the schema.
brk(b)  <> -- enter the break package.
bt      <col> -- add tag and its def for the whole col.
cc      <col_from col_after col_name> -- copy_column.
chgd    <col default> -- change_column_default.
chgvart <var type etc> -- change the type in declara.
clr     <> -- clear the table for <table_name>.
cls     <> -- clear the screen.
cr      <row_from row_after row_name> -- copy_row.
cw      <col width> -- change_column_width.
dc      <column> -- delete a column.
dr      <row> -- delete a row.
dt      <col tag> -- remove a tag and its def of a col.
ed      <> -- select an in core fsm to edit
exit    <> -- clear screen, exit to unix.
fc      <col new_tags> -- new tags for the selected col.
fr      <row new_tags> -- new tags for the selected row.
fs      <row col new_tag> -- new tag for slot (row col).
funl    <> -- list internal functions.
funa    <> -- add a function to the internal functions.
fund    <> -- del a function from the internal functions.
funs    <> -- sort the internal functions.
gtag    <row col> -- get_tag and its def of (row column).
ld      <fsm> -- load a new fsm from disk.
ldl     <fsm> -- attach a fsm in memory to the ste.
ls      <> -- print the schema part of the state_table.
lt      <col> -- a list of tags defined for this column.
ltu     <col> -- a list of tags used for this column.
m       <> -- display command menu.
mc      <col_from col_after> -- move a column.
mml     <> -- list multi_match functions.
mma     <> -- add a multi_match function.
mmd     <> -- delete a multi_match function.
mmr     <> -- resequence multi_match functions.
mms     <> -- sort multi_match functions.
mr      <row_from row_after> -- move a row.
n       <new_table_name> -- change working table name.
new     <new_fsm_name> -- create a new base fsm to start.
nml     <> -- list no_match functions.
nma     <> -- add a no_match function.
nmd     <> -- delete a no_match function.
nmr     <> -- resequence no_match functions.
```

```
nms      <> -- sort no_match functions.
oc       <all args are prompted> -- open a new column.
or       <row row_name> -- open a row after <row>.
p        <> -- generate a state_table of the current fsm.
pa       <attribute col> -- display_attribute.
port     <port_number> -- select a window for display.
postl    <> -- print the postprocesses.
posta    <> -- add a new function to the postprocesses.
postd    <> -- delete a function from the postprocesses.
postr    <> -- resequence the postprocesses functions.
posts    <> -- sort the functions in the postprocesses.
prel     <> -- print the preprocesses.
prea     <> -- add a new function to the preprocesses.
pred     <> -- delete a function from the preprocesses.
prer     <> -- resequence the preprocesses functions.
pres     <> -- sort the functions in the preprocesses.
pr       <fsm etc (prompted)> -- pretty print the fsm.
prf      <filename> -- save fsm plist formatted to disk.
ptf      <filename> -- save state_table to disk.
q        <> -- quit the state-table editor
remvart  <var> -- remove a type of a variable.
rc       <col new_col_name> -- rename a column.
rr       <row new_row_name> -- rename a row.
rseq     <increment> -- resequence rows.
save     <file_name> -- save the current fsm to disk.
sc       <> -- select a list of columns to print.
sortst   <> -- sort table by row name.
stty     <> -- set terminal type to sun(default) or vt100.
svs      <> -- flip the VERIFY flag.
tpa      <> -- add a template of tag def for the fsm.
tpc      <> -- change an template of tag def for the fsm.
tpd      <> -- remove a template of tag def for the fsm.
tpl      <> -- list templates of tag def for the fsm.
tps      <> -- sort the templates the fsm.
ty       <file> -- list the disk file indicated.
va       <variable type etc> -- add a new variable.
vc       <var type etc> -- change an existing variable.
vd       <var> -- delete a variable from the declaration
vl       <> -- list variables in a table format.
vs       <> -- sort variables in the declarations.
vsw      <> -- display the current VERIFY flag.
```

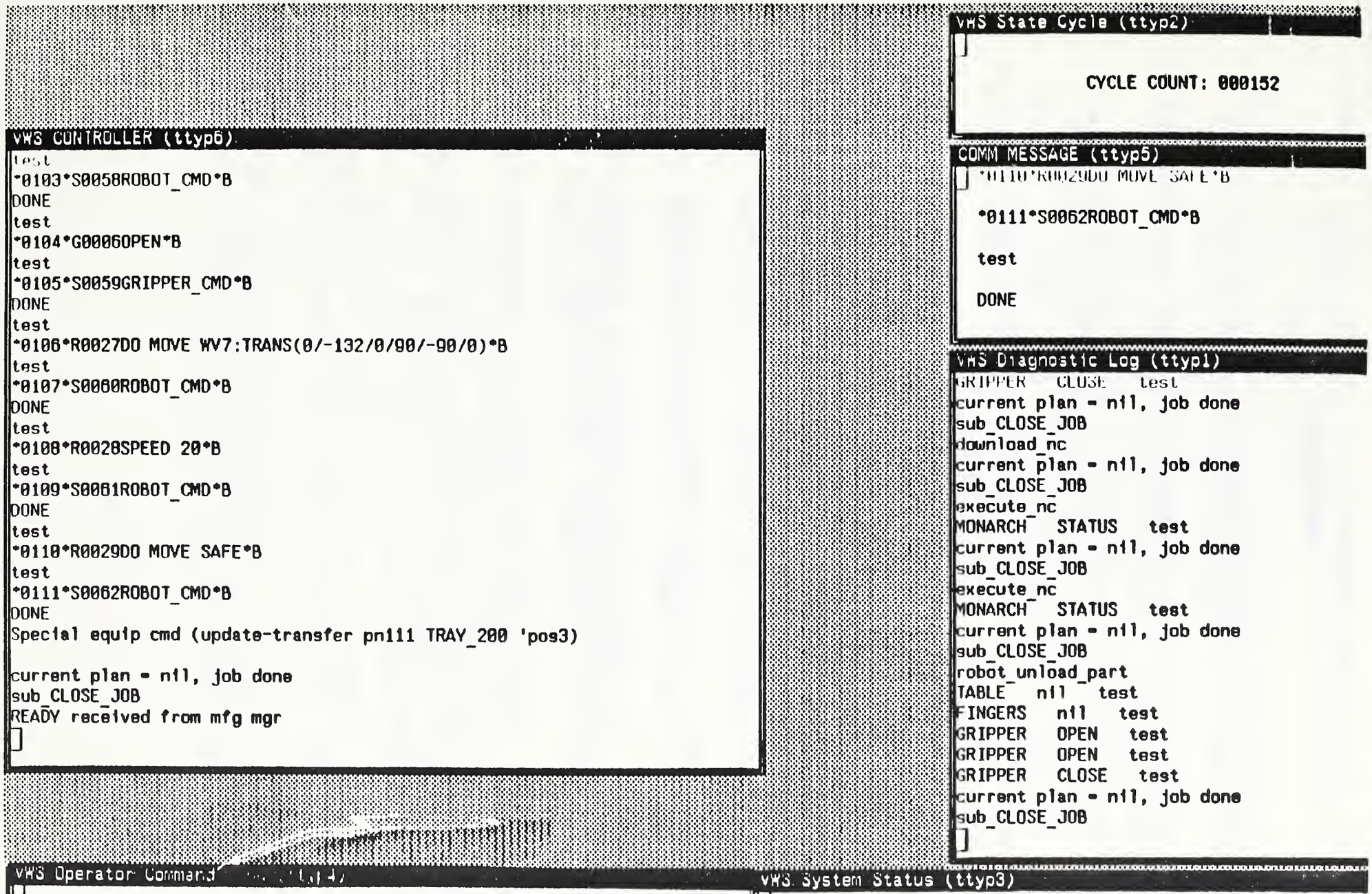
APPENDIX-4 A Listing of Work Elements Supported by the VWS.

vise_open
vise_close
gripper_open
gripper_close
clamp_pallet
release_pallet
download_nc
download_hp
download_ge
execute_nc
mtool_delete_nc
mtool_select_nc
mtool_cycle_start
use_nc
generate_nc
robot_load_part
robot_unload_part
lock_roller_table
unlock_roller_table
receive_tray
ship_tray
station_vacuum

APPENDIX-5 An Example of an Equipment Program

The following is an equipment program for the robot to pick up a part in the vise, flip the part to the other side, and put the part back in the vise.

```
 linked_to_prog    nil
 step              222
 prog  (prog 222   (ROBOT "SPEED 20")
             223   (STATUS "ROBOT_CMD")
             224   (ROBOT "DO MOVE SAFE")
             225   (STATUS "ROBOT_CMD")
             226   (GRIPPER "OPEN")
             227   (STATUS "GRIPPER_CMD")
             230   (ROBOT "DO MOVE WV1:TRANS(0/-25/0/90/-90/0)")
             231   (STATUS "ROBOT_CMD")
             232   (ROBOT "SPEED 10")
             233   (STATUS "ROBOT_CMD")
             234   (ROBOT "DO MOVE WV1:TRANS(0/-25/297/90/-90/0)")
            235   (STATUS "ROBOT_CMD")
             236   (GRIPPER "CLOSE")
             237   (STATUS "GRIPPER_CMD")
             238   (STATUS "GRIPPER")
             238a  (VISE "OPEN")
             238b  (STATUS "VISE_CMD")
             239   (ROBOT "DO MOVE WV1:TRANS(0/-25/0/90/-90/0)")
             240   (STATUS "ROBOT_CMD")
             241   (GRIPPER "ROTATE 180")
             242   (STATUS "GRIPPER")
             243   (ROBOT "SPEED 20")
             244   (STATUS "ROBOT_CMD")
             245   (ROBOT "DO MOVE WV1:TRANS(2/0/0/90/-90/0)")
             246   (STATUS "ROBOT_CMD")
             247   (ROBOT "SPEED 10")
             248   (STATUS "ROBOT_CMD")
             249   (ROBOT "DO MOVE WV1:TRANS(2/0/340/90/-90/0)")
             250   (STATUS "ROBOT_CMD")
             251   (GRIPPER "OPEN")
             252   (STATUS "GRIPPER_CMD")
             253   (ROBOT "DO MOVE WV1:TRANS(2/0/0/90/-90/0)")
             254   (STATUS "ROBOT_CMD")
             257   (ROBOT "SPEED 20")
             258   (STATUS "ROBOT_CMD")
             259   (ROBOT "DO MOVE SAFE")
             260   (STATUS "ROBOT_CMD")
             261   (GRIPPER "ROTATE 0")
             262   (STATUS "GRIPPER")
             262a  (VISE "CLOSE")
             262b  (STATUS "VISE_CMD"))
```


VWS State Cycle (ttyp2)
CYCLE COUNT: 000152
COMM MESSAGE (ttyp5)
*0111*S0062ROBOT_CMD*B
test
DONE
VWS Diagnostic Log (ttyp1)
GRIPPER CLOSE test
current plan = nil, job done
sub_CLOSE_JOB
download_nc
current plan = nil, job done
sub_CLOSE_JOB
execute_nc
MONARCH STATUS test
current plan = nil, job done
sub_CLOSE_JOB
execute_nc
MONARCH STATUS test
current plan = nil, job done
sub_CLOSE_JOB
robot_unload_part
TABLE nil test
FINGERS nil test
GRIPPER OPEN test
GRIPPER OPEN test
GRIPPER CLOSE test
current plan = nil, job done
sub_CLOSE_JOB
VWS CONTROLLER (ttyp6)
test
*0103*S0058ROBOT_CMD*B
DONE
test
*0104*G0006OPEN*B
test
*0105*S0059GRIPPER_CMD*B
DONE
test
*0106*R0027DO MOVE WV7:TRANS(0/-132/0/90/-90/0)*B
test
*0107*S0060ROBOT_CMD*B
DONE
test
*0108*R0028SPEED 20*B
test
*0109*S0061ROBOT_CMD*B
DONE
test
*0110*R0029DO MOVE SAFE*B
test
*0111*S0062ROBOT_CMD*B
DONE
Special equip cmd (update-transfer pn111 TRAY_200 'pos3)
current plan = nil, job done
sub_CLOSE_JOB
READY received from mfg mgr
VWS Operator Command
l-Refresh Menu
a-Attach
b-Break
d-Detach
dbl-Lcl Dmode
dm-Dpy Model
f-Flags
gc-Gbg Coll
lcm-Lcl Cmode
ldm-Lcl Dmode
m-Machine Lot
p-Peek
qlcm-quick lcm
qrcm-quick rcm
rcm-Rmt Cmode
rdm-Rmt Dmode
rt-ReceiveTray
s-Shutdown
st-Ship Tray
t-Test & Maint
x-Exit
VWS System Status (ttyp3)
tstate
mstate
cmode
dmode
READY
READY
LOCAL
LOCAL
detached subs
synched subs
attached subs
nil
nil
nil

REFERENCES

[Ref1] Albus,J. A.; AMRF Architectural Principles", to be published as an NISTIR, 1988.

[Ref2] McLean, C. R., "AMRF Operational Architectures", to be published as an NISTIR, 1988.

[Ref3] Lovett, Denver; "The Vertical Workstation's Equipment Controller of the Automated Manufacturing Research Facility", NBSIR 88-3769, April 21, 1988.

[Ref4] McLean, C. R., "Principles of the Cell Control System", to be published as an NISTIR, 1988.

[Ref5] Jun, Jau-shi; McLean, Charles R.; "Control of an Automated Machining Workstation", IEEE CONTROL SYSTEMS Magazine, Special Issue on Robotics and Automation, Vol.8 No.1, Feb. 1988. pp 26-30.

[Ref6] Nakpalohpo, Ibrahim; Jun. Jau-Shi; "Automated Equipment Program Generator and Execution System of the AMRF Vertical Machining Workstation" - to be published as NISTIR, 1989.

[Ref7] Rybczynski, S. et al., "AMRF Network Communications", NBSIR 88-3816, 1988.

[Ref8] Kramer, Thomas R.; "Data Handling in the Vertical Workstation of the AMRF at the National Institute of Standards and Technology", NBSIR 88-3763, April 21, 1988.

[Ref9] Brown, Peter F. and Ray, Steven R.; "Process Planning, System Architecture", NBSIR 88-3828, 1988.

[Ref10] Unger, Mark B. and Ray, Steve R.; "Feature-Based Process Planning in the AMRF", to be presented at ASME Computer in Engineering Conference", July, 1988.

[Ref11] Kramer, Thomas R.; and Jun, Jau-Shi; "The Design Protocol, Part Design Editor, and Geometry Library of the Vertical Workstation", NBSIR 88-3717, January 28,1988.

[Ref12] Kramer, Thomas R.; "Process Plan Expression, Generation, and Enhancement for the Vertical Workstation Milling Machine", NBSIR 87-3678, November 19, 1987.

[Ref13] Kramer, Thomas R. and Weaver, Rebecca E.; "The Data Execution Module of the Vertical Workstation", NBSIR 88-3704, January 6,1988.

[Ref14] O'Halloran, D. R. and Reynolds P. F., "A Model for AMRF Initialization, Restart, Reconfiguration, and Shutdown", NBS/GCR 88-546, May 23, 1986.

[Ref15] Libes, Don, and Barkmeyer, Ed., "The integrated manufacturing data administration system (IMDAS) -- an overview", Int. J. Computer Integrated Manufacturing, Vol.1, No. 1, pp. 44-49.

[Ref16] Furlani et al. "The Integrated Manufacturing Data Administration System (IMDAS)", to be published as an NISTIR, 1989.

[Ref17] Mclean, C., "The Vertical Machining Workstation of the AMRF: software Integration". Proceedings of the 1986 ASME Symposium on Intelligent and Integrated Manufacturing, Anaheim, CA. December 1986.

[Ref18] Magrab, E., "The Vertical Machining Workstation of the AMRF: Equipment Integration". Proceedings of the 1986 ASME Symposium on Intelligent and Integrated Manufacturing, Anaheim, CA. December 1986.

[Ref19] Winston, P. H. and Horn, B. K. P., "LISP", Addison-Wesley Publishing Company, 2nd Edition.

[Ref20] Albus, J., Barbera, A., and Negal, N., "Theory and Practice of Hierarchical Control", Proceedings of the 23th IEEE Computer Society International Conference, September, 1981.

[Ref21] Barbera, A., Fitzgerald. M. and Albus, J., "Concepts for a Real-Time Sensory-Interactive Control System Architecture", Proceedings of the 14th Southeastern Symposium on System Theory, April, 1982.

[Ref22] GEK-25384E, Mark Century 2000, Part Programming Manual, 2000MC CNC, May 1984.

READER COMMENT FORM

Document Title THE VERTICAL MACHINING WORKSTATION SYSTEMS

This document is one in a series of publications which document research done at the National Institute of Standards and Technology' Automated Manufacturing Research Facility from 1981 through March, 1987.

You may use this form to comment on the technical content or organization of this document or to contribute suggested editorial changes.

Comments:

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

If you wish a reply, give your name, company, and complete mailing address:

_____

_____

_____

What is your occupation?

_____

Note: This form may not be used to order additional copies of this document or other documents in the series. Copies of AMRF documents are available from NTIS.

Please mail your comments to: AMRF Program Manager
National Institute of Standards
and Technology
Building 220, Room B-111
Gaithersburg, MD 20899

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. **BIBLIOGRAPHIC DATA SHEET** *(See instructions)*	1. PUBLICATION OR REPORT NO. NISTIR 88-3890	2. Performing Organ. Report No.	3. Publication Date NOVEMBER 1988

4. TITLE AND SUBTITLE

The Vertical Machining Workstation Systems

5. AUTHOR(S)

Jau-Shi Jun

6. PERFORMING ORGANIZATION *(If joint or other than NBS, see instructions)*

NATIONAL BUREAU OF STANDARDS
U.S. DEPARTMENT OF COMMERCE
GAITHERSBURG, MD 20899

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS *(Street, City, State, ZIP)*

National Bureau of Standards
Gaithersburg, Maryland 20899

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)*

The document presents an overview of the Vertical Machining Workstation (VWS) in the Automated Manufacturing Research Facility (AMRF) at the National Bureau of Standards. It gives details of the workstation control flow. This prototype system demonstrates flexible computer integrated manufacturing for a family of prismatic parts. The workstation software components include: a feature-based design system for defining part geometries, an automatic process planning and numerical control code generation system, an automatic robot program generator, a state machine-based hierarchical control system which executes process plans, a diagnostic tools package, and mailbox communications software. The system is capable of running stand-alone, as a single station manufacturing system, or integrated under the AMRF cell controller.

12. KEY WORDS *(Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)*

Automated Manufacturing Fesearch Facility (AMRF); manufacturing automation research, factory control architectures, vertical machining, process planning, interface standards, automated nc generation, feature-based design

13. AVAILABILITY

☒ Unlimited
☐ For Official Distribution. Do Not Release to NTIS
☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.
☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

73

15. Price

$13.95

Jun 13, 2016