MOSAIC
SYSTEM DESCRIPTION, SPECIFICATION AND PLANNING

Revision 1.1
6/15/89

Designed and Implemented by:

Israel Greenfeld
Fred B. Hansen
Jim Fehlinger
Louie Pavlakos

Robotics Research Laboratory
Computer Science Division
Courant Institute of Mathematical Sciences
New York University
719 Broadway 12th Floor
New York, NY 10003

ABSTRACT

The MOSAIC system is specified and described. MOSAIC is an open-system controller intended for use on the next generation of machine tools. Its configuration is based on the open-system computer technology, thus providing a suitable solution for the multi-device multi-sensor environment of future machine-tools. The implementation of the system at the Robotics and Manufacturing Laboratory of New York University is described. The current configuration is based on a Sun workstation, an extended VMEbus, and a multi-processor environment. The system is demonstrated on a 3-axis milling machine, equipped with subsidiary devices like a manipulator and a vision system. The document contains information on the current configuration, future planned enhancements, and various preliminary ideas. Details of the hardware and software are included. Our short-term and long-term plans for research and implemetation are also presented.

CONTENTS

1. INTRODUCTION

This document specifies and describes the **MOSAIC** system, developed at the Robotics and Manufacturing Laboratory at NYU. MOSAIC is an advanced *open-system* controller intended for use on the highly *self-sustaining* next generation machine tools. The MOSAIC system is based on the current open-architecture computer technology. The controller is conceived as a wide-scope system, responsible for highly automated design, planning and production. The MOSAIC concept is currently being implemented at NYU on a milling machine environment, and named the MOSAIC Project. The concept, though, is extendable to any complex machine environment.

A self-sustaining machine tool is equipped with dextrous manipulators dedicated for the continuous needs of machining, such as chip clearance and part relocation. A variety of sensors provide vision, touch, force, noise and temperature senses, with the tasks of recognizing events, in-cycle inspection and optimizing the machining parameters. A rich supporting design environment is essential, including:- cadcam for part and tool path design; expert systems and libraries of technical information for an optimal and efficient design; and an operating environment with process planning capability. These features are captured in figure 1.1.

As an open-system, a machine tool is equipped with a general purpose computer, such as a PC or a workstation, enhanced with additional standard computer hardware, that controls the axes of motion and the various devices, and manages programs and data. Communication utilizes networks that are universally accepted in the computer world, and thus resources may be shared as needed. The machine is adaptable to the changing environment and tasks, in terms of its controller's computer configuration, and in terms of the mechanical construction.

The MOSAIC controller is specified and designed especially for the projected self-sustaining open-system future environment of machine tools. The concept (and it NYU implementation - in brackets) is based on a general purpose computer (a Sun workstation), running a standard operating system (Unix), programmed in a standard programming language (C), and having a standard hardware (VMEbus). It provides a flexible environment that can drive the machine, the manipulators and the sensors. It can be configured as a controller for a single machine or for multiple machines. It uses the most advanced features of today's computer technology, such as high and expandable computational power, graphic user interface, and support of a variety of communication networks. It benefits from the huge market of hardware and software of standard computer equipment provided by third party vendors. Also, with the current trends in the general purpose computer industry, the controller is expected to have a good price-performance ratio.

The specification and design of the controller involve a variety of hardware and software technologies. The research is intended to investigate the broad implications of such a controller on machining languages and operating systems, as well as on integration with machine tools and devices. Currently, three main research and development areas associated with this open-architecture system have been identified and will be gradually incorporated in the system:-

i) The integration of the *Open-System Machine Tool Controller*. It includes the integration of a general purpose computer and a machine tool with new devices and sensors. These devices are part of the machine-specific configuration, and are controlled locally by the open-system controller. It is emphasized that no special computer hardware has to be developed, since only general purpose equipment is used. It is expected that with the future introduction of more powerful processors and workstations, and the development and commercialization of parallel and distributed-processing computer systems those architectures may gradually substitute elements of the current architecture. The *openness*, though, is kept.

ii) The development of a *Real Time Operating System for Manufacturing*, suitable for the very high speed control required for machining and manipulation operations. Current general

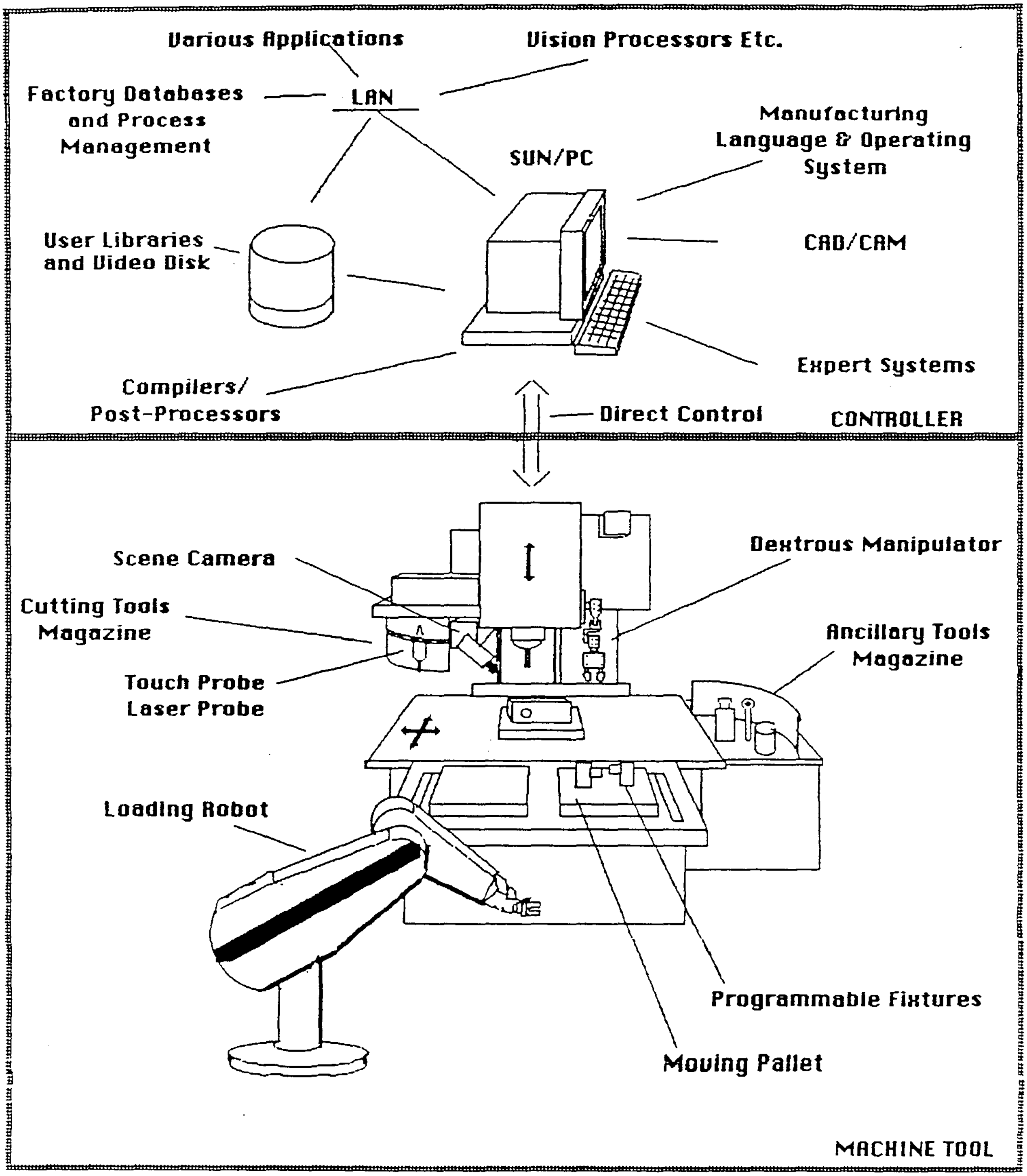


Figure 1.1. Projected Machine Environment

purpose real time operating systems do not provide the response time essential for maintaining the speed, accuracy and safety features inherent to machine tools. Typically, an update rate of all degrees of freedom is expected to be around one milisecond, thus allowing for high speed machining and manipulation. The operating system should interface to industry standard operating systems such as Unix or OS/2, which provide high level management, filesystem operations, communications, and a good programming environment.

iii) The development of an *Advanced Manufacturing Language* for manufacturing programming, that resides in the open-system controller. While existing languages such as APT for machining, ladder logic for programmable controllers, AML for manipulating, and CML for coordinating various operations should be supported, a more universal and flexible language is needed. It includes provisions for real time control needed for the operation of accessory devices in conjunction with the machining process and other manufacturing processes, a more direct connection to cadcam systems, and a flexible interface for user applications. Artificial intelligence techniques are to be used for the generation of process plans, and for adaptive control.

The following specification describes the MOSAIC system and the requirements from the software to be developed for the MOSAIC project. Also included are our short-term and long-term plans for research and implementation over the next three years. The specification is written with some generality in mind, so that we may retain flexibility and universality as the needs evolve in time. Also, provisions are taken to ensure that introducing additional new equipment in the future, such as motion controllers from other manufacturers or digital-analog I/O boards, will benefit as much as possible from the software already developed.

The document is intended for use by the system developers, system operators, and those who are interested in the details of the NYU project. Other documents, listed ahead, contain a more general information. Since this is a work in progress, the document contains a mixture of current configuration details, future enhancements, and even preliminary ideas. The system developers should be consulted for specific details on the actual state of the project.

2. APPLICABLE DOCUMENTS

The following references are applicable for this document:

2.1. Introductory Documents

- Papers and proposals describing self-sustaining machine tools and the MOSAIC concept, specification and implementation [1-5].
- Related projects on open-system machine controllers [6-8].
- Books on intelligent manufacturing, part programming, Computer Numerical Control (CNC), Programmable Logic Controllers (PLC), and motion control [9-14].
- Papers on real-time operating systems, advanced manufacturing languages, expert systems and programmable fixtures [15-19].
- References on object-oriented programming languages [20-22].

2.2. MOSAIC System Manuals

- Dedicated-hardware description, including milling machine, motors, Sun computer hardware, and VMEbus hardware [23-30].
- Dedicated-software description, including Unix, the Sage operating system, and the Hierarchical Control system (HIC) [31-33].

2.3. Other Related Documents

- Handbooks and computerized libraries of engineering data [34-37].
- Software packages, including cadcam, machining programming, and symbolic computing [38-41].
- Standards for parts' definition and communications, including VMEbus [42-44].
- Operating manuals of related manufacturing equipment [45-50].

3. SYSTEM CONFIGURATION

3.1 The Generic Controller

A generic multi-machine configuration of an open-system controller concept is presented in figure 3.1. It consists of a *Plant Controller*, a few *Machine Controllers*, and machines. A *Local Area Network (LAN)* establishes communication between the plant controller and the machine controllers, and with other plant controllers in the factory. The machine controllers are directly connected with the machines and the devices and sensors mounted on them. They execute the real-time control operations such as machine-tool axes' motion, manipulators' motion, part clamping, sensor data acquisition, and vision. The plant controller is dedicated for the supervision of a few machine controllers, serves as an operator's terminal, and provides file systems and local and wide area communication services. It also serves as a design and programming workstation.

Both controllers use an industry standard data bus such as the VMEbus, the PCbus or the Multibus. They communicate over a LAN such as MAP/TOP and EtherNet (both typically yield appx. 10 Mbits/sec) or the expected FDDI fiber-optic LAN (estimated 100 Mbits/sec). For single machine configurations, a direct connection of the two buses may be obtained through a bus repeater, thus providing a very fast data transfer rate (yields appx. 300 Mbits/sec). All these data transfer rates are theoretical maximum values. Practical speeds may be up to four times slower, depending on the physical configuration and the nature of the line's traffic. Usually, direct bus connections, which limit the distance between the two controllers to a few feet, will not be necessary, since the real-time work is done mainly on the machine controller. Exceptions to that are development systems, such as the current MOSAIC implementation, where an extended bus provides higher flexibility. The design of the system is such that this variation in the bus configuration is transparent to the user.

Communications on both sections of the bus are supported by communication processors that relieve the load from the real-time control processors, by performing file transfer, data compression, command interpreting, etc. The real-time computations are performed by processors running a real-time operating system, and executing the user application programs. Current processors are based on microprocessor technology such as the 68020. Specific operations such as motion cotrol, image processing, and various I/O operations are performed by dedicated processing systems. Workstation services, like file systems and terminals, are provided by a general purpose computer such as a Sun or a PC, supporting the variety of compilers, application programs and utilities available on the market. Some of the services may also be provided by an auxiliary workstation hooked to the machine controller, located at a site close to the machine.

The current MOSAIC implementation is a variation on this configuration. Its description is contained in the following sections.

3.2. System Contents

The current system configuration includes the following main sub-systems:

3.2.1. Hardware List

(1) Intermark Hartford SP-1 Machine Tool with a T-Ram.

(2) General Motion Motors: three 44 in-lbs axis motors with drivers, and one 5 HP spindle motor with driver.

(3) Electrical Cabinet (designed and built at NYU).

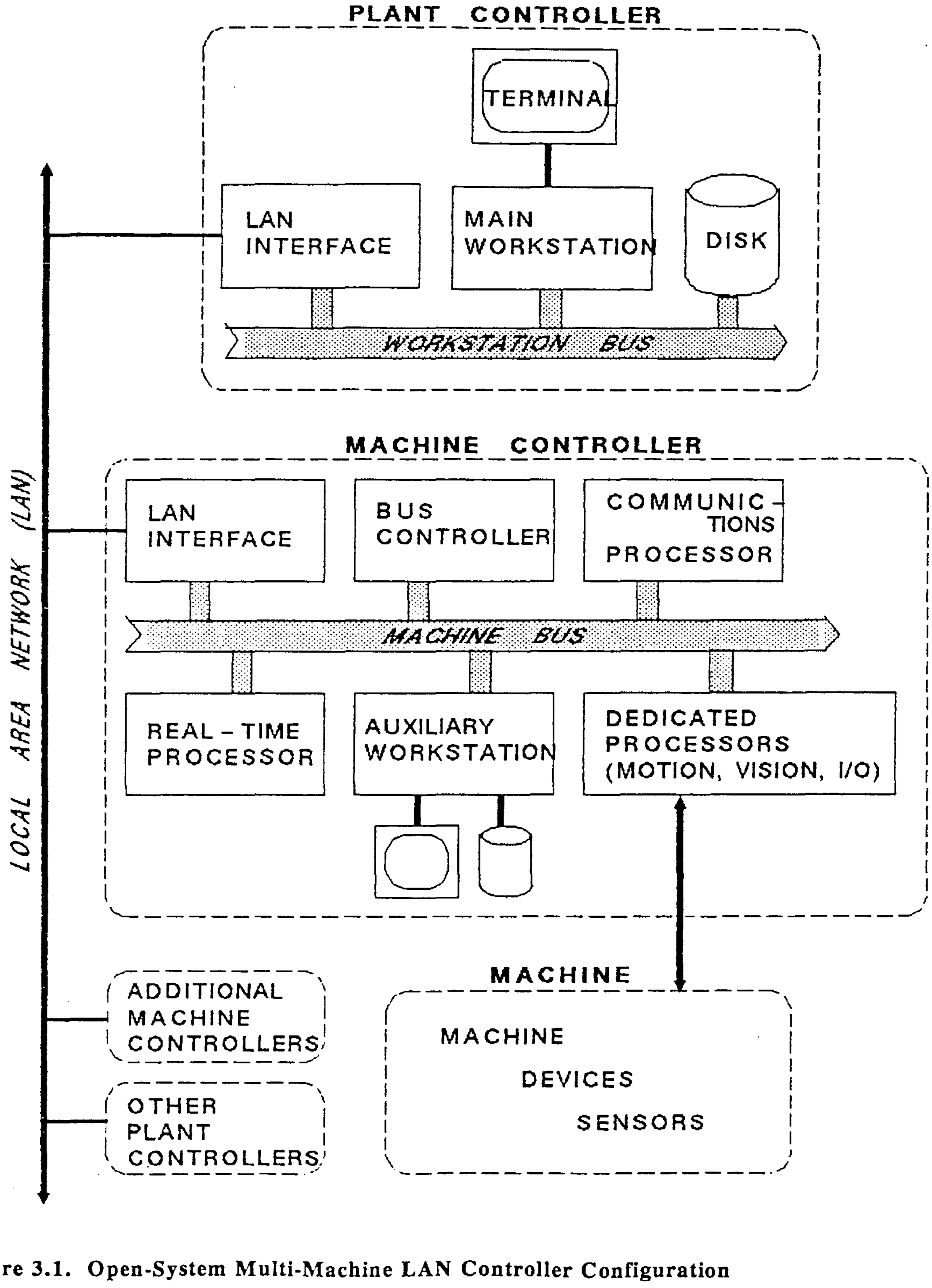


Figure 3.1. Open-System Multi-Machine LAN Controller Configuration

(4) Sun Microsystems 3/160M Workstation with monochrome monitor, diskless.

(5) BICC-Vero VMEbus Microrack with a J2 backplane.

(6) HVE Engineering VMEbus Repeater 2000 with cables.

(7) Pacific Microcomputers PV682 MC68020 16 Mhz MPU with a MC68881 FPU.

(8) 2 Creonics VME-EG0 Motion Controller boards with a gearing option.

(9) Various cables and accessories.

3.2.2. Software List

(1) System software - command interpreters, compilers, window environments, etc. (under Sun's Unix).

(2) The SAGE Real Time Operating System.

(3) HIC Hierarchical Control System.

(4) User application software - motion control and planning, manufacturing languages, process planning, descrete events handling, etc.

(5) The Machinist Expert System.

(6) MCS Anvil-5000 Cadcam system.

(7) Users' libraries - Anvil's Grapl, CarrLane Tools and Fixtures, Video Disk, etc.

3.3. Hardware Configuration

Figure 3.2 shows an overview of the MOSAIC system configuration. It describes the complete system, which is currently in its first phase of construction. This configuration is slightly different from the generic configuration presented in figure 3.1. The plant controller is termed *supervisory controller* due to the single-machine configuration. The VMEbus is repeated directly rather than through a LAN, but Ethernet is supported just the same by the real-time system.

The milling machine, a SP-1 from Intermark Hartford, has 3 travel axes and a spindle, all using AC induction motors and encoders from General Motion. The system uses a Sun 3/160 workstation with a 19" graphic monitor. The Sun's 9U VMEbus is extended to a standard 6U VMEbus through the VMEbus Repeater 2000 from HVE. The Sun provides file system and communication services, including a connection to the lab's Ethernet. This enables the access to the vast variety of other computer services available in the lab, including technical and programming libraries, the Anvil-5000 Cadcam system, a Vicom image processor, a WORM video-disk, and more.

The extended VMEbus rack, from Bicc-Vero, hosts a Pacific-Microcomputer 68020 CPU board with a 68881 FPU, which runs the real-time operating system and the application programs. Dual-axis VME-EG0 Motion Controllers from Creonics are used for controlling the machine's 3 axes and spindle, all with full position and velocity control, using a PIVF (Proportional, Integral, Velocity Feedback and Velocity Feedforward) digital control scheme. Both the Sun and the VMEbus environment will be enhaced with additional mass storage and memory. New processor architectures (68030, RISC, etc.) will be used as applicable.

Additional motion controllers are planned for the operation of the Dexman manipulator. A real-time image processor board-set is planned for monitoring, inspection and manipulator guidance. Currently, the video cameras are connected to a remote Vicom vision processor. The video data will also be projected onto a window on the Sun's screen. Programmable fixtures and various sensors and devices are also planned. This device environment is shown in figure 3.3.

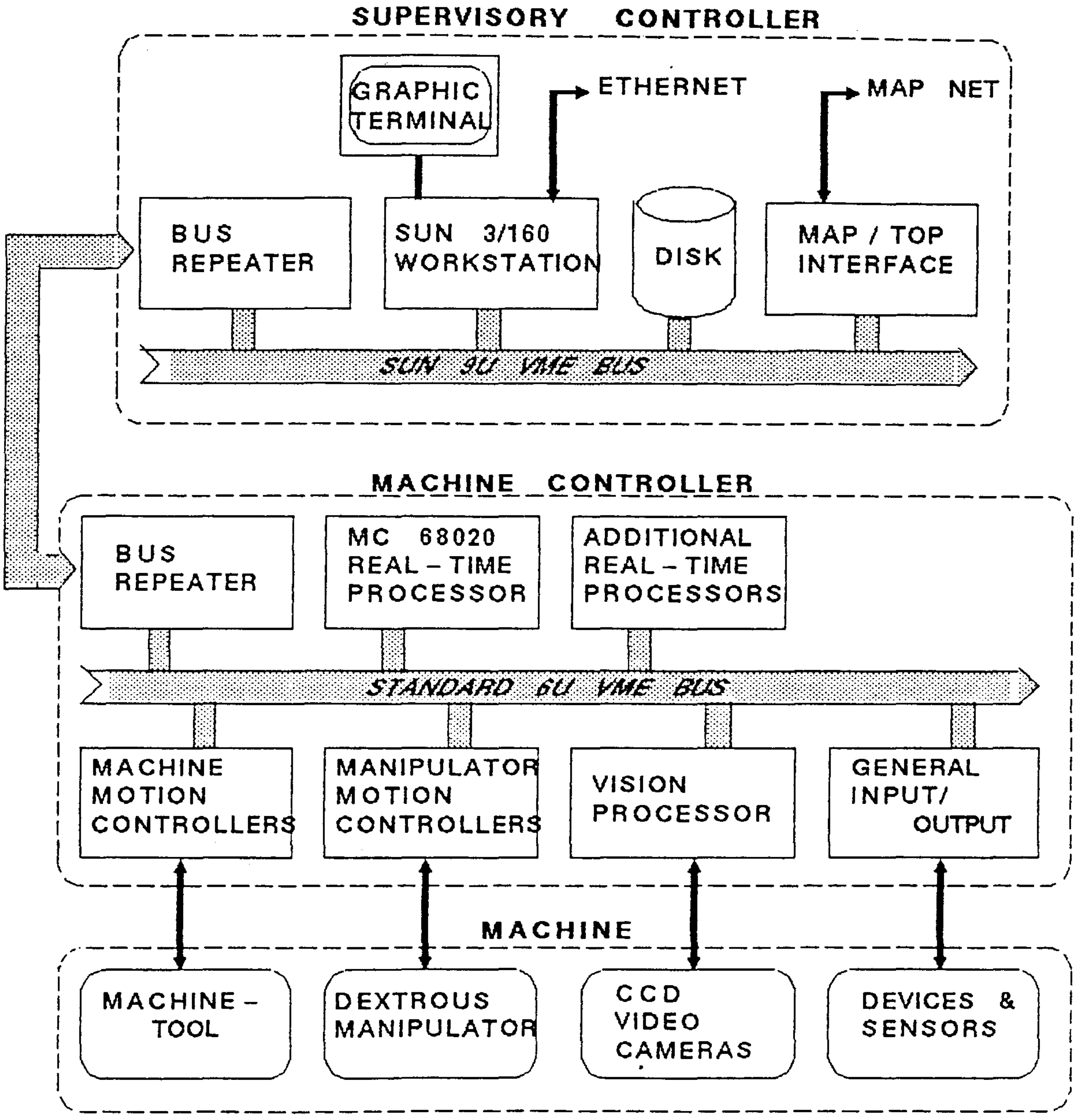


Figure 3.2. The Current Single-Machine VMEbus Configuration

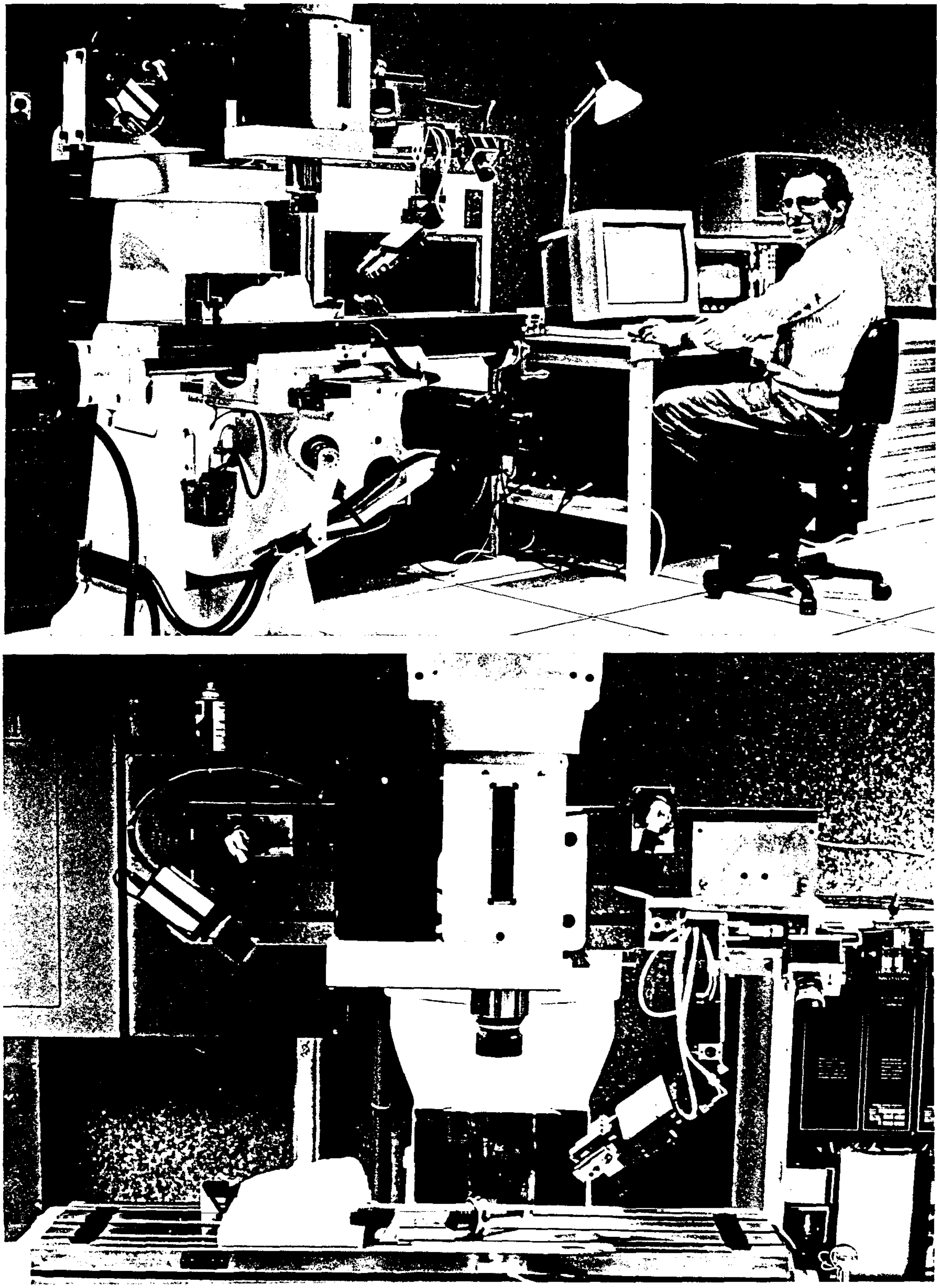


Figure 3.3. Machine and Devices Environment

Figure 3.4 shows the current equipment layout in the lab. The computer system is remote from the machine itself, and the computer terminal - monitor and keyboard - serves as the machine operator's panel. Video cameras provide a continuous monitoring of the cutting scene. All the system's components were purchased off-vendors'-shelf. The electrical connections were all designed and done in-house to retain flexibility in integration.

3.4. Software Configuration

Figure 3.5 shows the MOSAIC software configuration, current and planned. Details of the current capabilities are included in the *Software* section. Five categories are specified, including, from the top level to the bottom level: a graphical user interface; languages; planning applications; control applications; and machine applications.

The low level **machine applications** are mainly vendors' proprietary software running on dedicated motion controllers, vision processors, etc. The system provides interfaces to these sub-systems, unique to each sub-system. These interfaces include communications over the VMEbus, parameters' setting, interrupt handling, etc.

The **control applications** software provides motion control, from the single-axis control, to linear, circular and spline interpolations (multi-axis coordinated motion). It also provides control of the analog and descrete devices and sensors, like programmable fixtures, force sensors, vision, etc.

The **planning applications** include motion planning for cutting complete surfaces. This feature is currently available only as an off-line tool-path generation by the Anvil-5000 Cadcam system. Future research, involving an expert system for machining (Machinist) and computational geometry techniques, may provide a process level of motion planning. We plan also to integrate the expert system and the Cadcam system with the controller to provide quasi-real-time adaptive control of operations. A closed loop force control is now being implemented by measuring motor torques indirectly (in current mode) and varying the travel feedrate and spindle speed on the fly.

On the **languages** level, we currently have an interpreter for APT. An interpreter for AML will be incorporated with the introduction of the manipulator. The interpretation is done in real-time, eliminating the need for the traditional post-processing. We plan to concentrate efforts on the introduction of a universal manufacturing language to enable us to control the machine, the manipulator, the vision system, and the other sensors and devices, concurrently.

On the top level, we have a graphical **user interface**, consisting of three interaction screens: the Designer Screen, the Machine Operator Screen, and the Plant Operator Screen. Each screen uses a few application windows (operations, monitors, diagnostics, cadcam, programming, planning, etc.). The graphics is Sun/Mac style, using windows, icons, dialogue boxes, and pull-down menus. Currently, as a first phase, we have parts of the operator's screen, implemented experimentally under an X-Window environment.

The **Operating Systems** interact with all the above-mentioned levels. The higher level software is generally serviced by the Unix operating system running on the Sun. Most of the operating system design, and the application programs, were developed in C under Unix. The lower level software is supported by the Sage real-time operating system, developed in-house for robotic applications and adapted for MOSAIC. A shell (command interpreter), named the MOSAIC shell or Conch, is running on Sage, and supports the high level commands which operate the system, including communications with the motion controllers, motion commands, status inquiries, etc. A file server is running on the Sun and provides file system support for Sage. Commands generated by the user on the graphic interface are piped through the file server to the MOSAIC shell for execution. A Hierarchical Control System (HIC), also developed for robotic applications, provides an extensive library of C routines

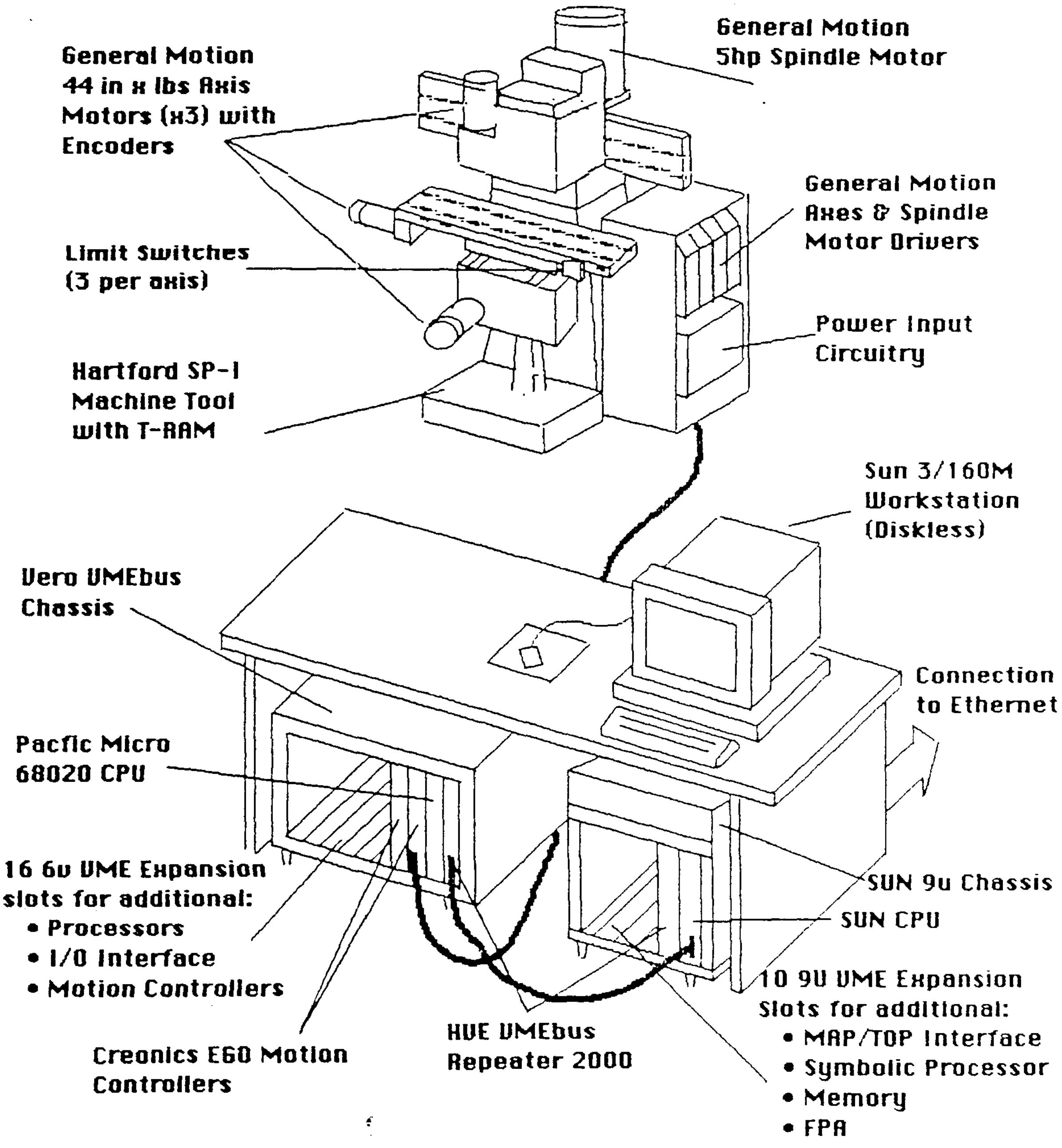
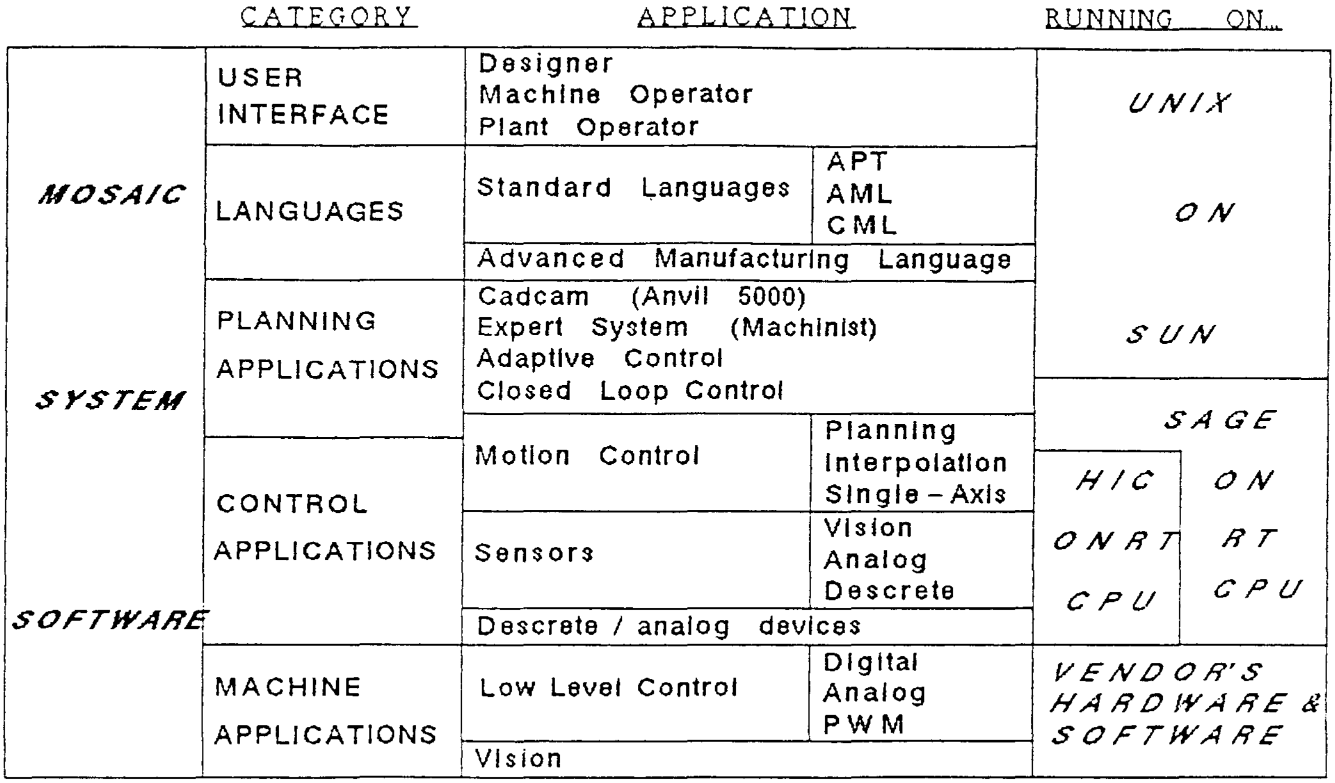


Figure 3.4. Current Equipment Layout



	CATEGORY	APPLICATION		RUNNING ON...	
MOSAIC SYSTEM SOFTWARE	USER INTERFACE	Designer Machine Operator Plant Operator		UNIX ON SUN	
	LANGUAGES	Standard Languages	APT AML CML		
		Advanced Manufacturing Language			
	PLANNING APPLICATIONS	Cadcam (Anvil 5000) Expert System (Machinist) Adaptive Control Closed Loop Control			
	CONTROL APPLICATIONS	Motion Control	Planning Interpolation Single – Axis	SAGE H/C ON RT CPU	ON RT CPU
		Sensors	Vision Analog Descrete		
		Descrete / analog devices			
	MACHINE APPLICATIONS	Low Level Control	Digital Analog PWM	VENDOR'S HARDWARE & SOFTWARE	
		Vision			

RT = Real Time
PWM = Pulse Width Modulation

Figure 3.5. Software Configuration

for real-time control of operations, starting from basic operations (e.g. single axis motion), up to object level operations (e.g. interpolations and motion planning). The real-time operating system will be enhanced to support multi-processing and new advanced processors like the MC 68030 and RISC architecture.

4. HARDWARE

This section contains the details of the current MOSAIC hardware configuration. It also includes a brief description of the planned near-future enhancements.

4.1. Main Equipment

The main sub-systems and components are described ahead:

4.1.1. Hartford SP-1 Machine Tool

A three axis plus spindle knee-type small machine tool. The X and Y axes are direct drives through ball screws. The Z axis is driving a ball screw through a 1:1 ratio timing belt. The spindle is driven through a 1:1 ratio timing belt, to enable servo-positioning as well as velocity control. All ball screws have a pitch of 0.2 inch/revolution. Each axis is equipped with 3 limit switches: Home, +overtravel and -overtravel. The machine has a large T-Ram which allows for spindle head sliding and addition/replacement of spindle heads.

Ball screws pitch:	0.2 ipr (inch per revolution)
Accuracy:	est. 0.001 inch per feet per axis
Repeatability:	est. 0.0005 inch per axis
Maximum axis velocity:	est. 200 ipm (inch per minute)
Maximum spindle velocity:	4000 rpm. Currently limited to 2000 rpm due to excessive vibrations

Travels:	X (longitudinal) 24"
	Y (lateral) 11"
	Z (vertical) 5"
	Knee (vertical-manual) 16.5"
	T-Ram (lateral-manual) 12.5"
Max. load:	500 lbs

4.1.2. General Motion Motors

3 three-phase AC induction axis servo motors and a spindle motor with a matching driver for each motor. A shared power supply supplies 325 VDC to all drivers. These motors are controlled by a 3-phase PWM (pulse width modulation) generated by the drivers. They maintain a full torque at stall and at low velocity (up to 1000 rpm appx.), and a full horsepower at higher speeds. Each motor is equipped with an overtemperature thermostat.

Each axis motor is equipped with a 2000 ppr (pulse per revolution) 6 channel encoder. The spindle has a 1000 ppr encoder. Encoder channels include two 90 deg phase-shifted channels, a marker channel (single pulse per revolution) and a complement channel for each of those. This enables the use of a 4X quadrature mode, thus attaining a 8000 ppr resolution (4000 on spindle).

The drivers are controlled by an external input signal (from the computer) between -10 VDC and +10 VDC. The signal, in our installation, is cofigured to produce a proportional torque (torque mode) at low velocity (up to 1000 rpm appx.) and a proportional horse power at higher velocities. The system may alternatively be configurated in velocity mode by replacing personality modules inside the drivers. The drivers are enabled by an external signal (from the computer) and send back a ready signal. By wiring they are connected to the limit switches so that they may not be enabled to produce torque in a violated direction.

Continuous torque:	axis 44 in-lbs
	spindle 211 in-lbs

Peak torque:	axis 150 in-lbs
	spindle est. 500 in-lbs
Maximum velocity:	6000 rpm for each motor.
Continuous power:	axis 1.2 HP
	spindle 5 HP (7.5 HP 30 min.)
Encoder resolution (4X quadrature):	axis 8000 ppr
	spindle 4000 ppr

4.1.3. Electrical Cabinet

The electrical cabinet contains the motor power supply and drivers, the power input circuitry, the emergency stop logic, and various computer to machine connections. The major components in the cabinet are a main circuit breaker, a main contactor, the motor drivers and power supply, a control power transformer, two DC power supplies, a delay relay, emergency and overtemperature relays, cabling and connections terminal blocks, and two cooling fans.

The main power input is 220 VAC 3-Phase protected by a 40 amps circuit breaker. The cabinet contains supplies of 110 VAC control power, 24 VDC relay logic power, and +15 VDC, -15 VDC and 5 VDC logic power.

The emergency stop responds to the emergency pushbuttons (one on the machine and another movable), the thermostats and the motion controllers' watchdog. Watchdog activation may be the result of an MCC (Motion Controller Card) failure, a VMEbus failure or a VMEbus reset. The emergency stop sequence starts with an attempt to stop all motion by the computer, followed by a main power cut-off after 0.2 sec (selectable).

All descrete logic signals are routed through a main terminals block section, and may be accessed for testing.

4.1.4. SUN 3/160M Workstation

The host computer is a SUN 3/160 monochrome 68020 workstation without disks. The chassis contains 12 9U VMEbus expansion slots. Two slots are currently occupied by the main CPU board (in the 1st slot) and the primary VMEbus repeater board (in the last slot). The free slots are intended for future expansion such as FPU, memory, disk controller, MAP/TOP communication etc.

4.1.5. Vero VMEbus Microrack and HVE Repeater

The VMEbus contains 20 6U VMEbus expansion slots. The rack includes its own power supply and cooling system. The J2 backplane has wirewrap pins on (the free) rows A and C, with matching connector shrouds on its reverse side, to mate with DIN 96 pin type cable connectors. These are used to connect to the MCC's, and possibly future boards that make use of this arrangement.

Currently the following slots are occupied: The secondary VMEbus repeater board (1st slot), the Pacific Micro MPU board (2nd and 3rd slots), and the two Creonics motion controllers (4th and 5th slots). The free slots are intended for future expansion such as additional motion controllers, processors and I/O interface boards. The SUN VMEbus and the microrack VMEbus are interconnected with a full 32-bit bus repeater, which uses a board on each bus and two ribbon cables.

4.1.6. Pacific Micro 68020 MPU

The CPU board is based on a Motorola MC68020 32-bit microprocessor running at clock rate of 16.67 Mhz. The board includes also a Motorola MC68881 floating-point math co-processor, an advanced memory management unit, 1 mbyte Dram memory, five programmable timers, two serial ports and multiple interrupt sources.

The MPU runs the real time operating systems SAGE and HIC, and controls the real time operations of the motion and I/O logic. Startup and user programs are loaded through a serial port or read over the VMEbus.

4.1.7. Creonics EG0 Motion Controllers

The two motion controllers handle the motion of axes X and Y (the 1st board) and axis Z and Spindle (the 2nd board). Each board contains a 80186 High Integration Microprocessor for motion control, and a proprietary CX2216 Custom Motion Control Peripheral IC for encoder input decoding and torque output command. The board supports a few I/O analog and descrete channels. All the I/O channels are opto-isolated. External power supplies (from the electrical cabinet) are provided to maintain the isolation integrety.

The MCC implements a PIVF (proportional, integral, velocity feedback, and velocity feedforward) digital control loop. The output control signal may be PWM or analog (our case). The MCC supports a variety of motion modes, including indexed motion with profiled velocity, track motion for coodinated axes motion, homing sequence, electronic gearing, and electronic cam (not purchased). Control loop parameters and gains may be set by software and read by the host, some on the fly.

Each MCC contains 4 descrete logic inputs: Home, +Overtravel, -Overtravel and Drive Fault (Drive Ready on the motor drivers). These inputs may be configurated to generate appropriate motion sequences on the corresponding axis, or as general purpose logic inputs. In our installation they will be configured as follows: For each travel axis, Home will be used for its intended use of homing the interrupted axis only, and the other 3 inputs will be configurated as general logic inputs to produce a motion stop of all axes. For the spindle, all 4 inputs are configurated as general inputs to produce a motion stop of all axes. The specific input assignments are detailed later in this document.

The MCC also provides a Drive Enable logic output which enables/disables each of the motors' drivers, and a watchdog relay contact (configurated as Normally Closed) which is wired in chain with the emergency stop string.

4.1.8. Additional Equipment

The following devices are to be mounted on the system in the future, and will be described in detail at a later stage. The detailed expansion plans are included in the *Program Plans* section of this document.

(1) Dexman - a dedicated machine-mounted 3-5 degrees of freedom manipulator, with matching motion controllers. See figure 4.1.

(2) Fixtures - a programmble fixturing system.

(3) Vision - a vision system with a CCD camera(s) and a VMEbus vision processor.

(4) Sensors - various sensors for measuring stress, force, temperature, noise, vibration etc., and matching I/O boards.

(5) Communications - Sun/VMEbus boards for factory communications such as the MAP protocol.

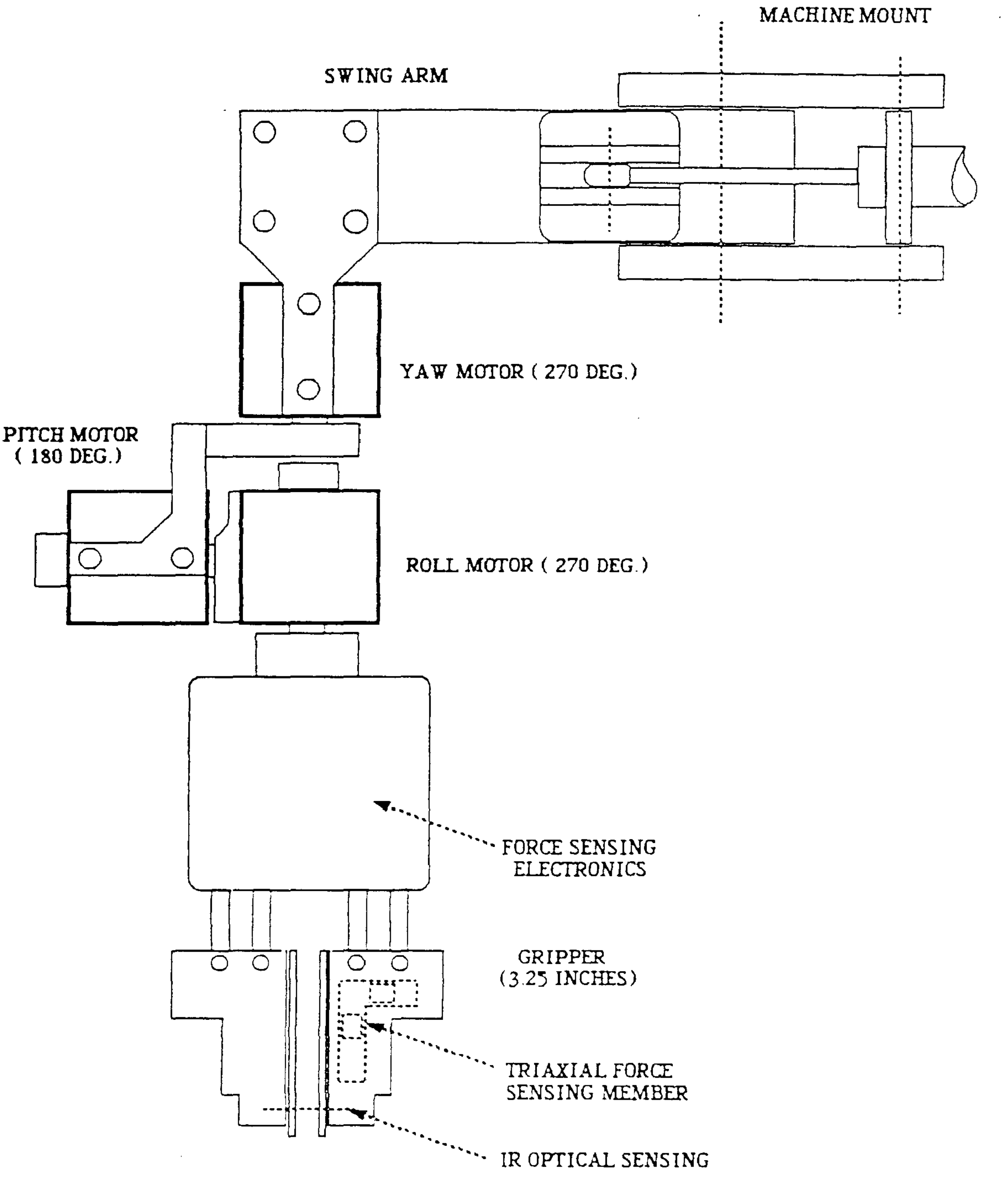


Figure 4.1. The Dextrous Manipulator (Dexman)

4.2. Wiring and Signals

The system wiring diagram, and the main cables and signals between the sub-systems, are described ahead. Additional detailed information on component placement and pin connections is contained in the appendix *Components Wiring Pin Connections.*

The following abbreviations are used in this section:

24,0,5,15	24 VDC,.0 VDC, 5 VDC and 15 VDC logic respectively
10	-10 VDC to +10 VDC analog
mac	machine
comp	computer
cab	cabinet
mcc	motion controller card
ls	limit switch
tstat	thermostat
tsr	thermostat switch relay
esr	emergency stop relay (or button)
mcr	main contactor relay's auxiliary contact
x y z s	XYZ axes and spindle respectively

4.2.1. Wiring Diagram

Refer to figure 4.2 for a full wiring diagram of the system.

Note: The watchdog chain shown on the diagram is currently deactivated (jumpered) because of the dependency of the MCCs on the 24 VDC supply from the electrical cabinet. This supply is used by the MCC to enable the watchdog. The problem may be rectified by adding a stand-alone 24 VDC power supply to the VMEbus microrack.

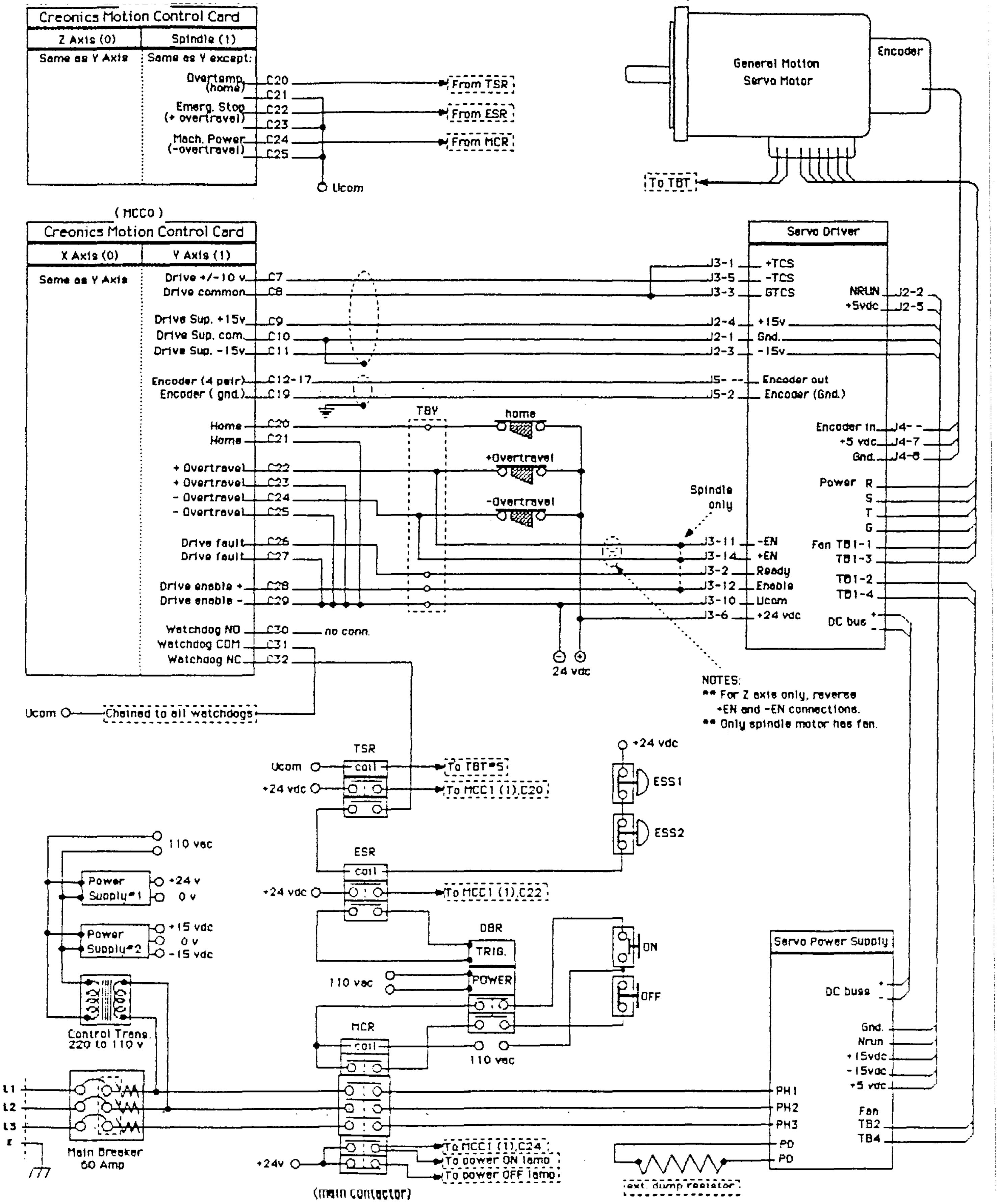


Figure 4.2. System Electrical Wiring Diagram

4.2.2. Main Cables

The following cables interconnect the main sub-systems:

Main Interconnection Cables			
Cable	Contents	From-To	Type
Spindle Motor	motor power, encoder, fan	cab-mac	conduit
X Y Motors	motor power, encoder	cab-mac	2 conduits
Z Motor	motor power, encoder, z limit switches	cab-mac	conduit
Limit Switches	x y limit switches	cab-mac	conduit
X Y Z S	logic signals and power, torque signal and power, encoder	cab-comp	4 braided
Repeater	VMEbus extension	SUN-chassis	2 ribbon

4.2.3. Main Signals

The following signals are sent between the computer, the machine and the electrical cabinet. VMEbus signals are excluded.

Main Inter-Systems Signals						
Signal	Type	Active	From	To	Axis-Device	Source
Home	24	0	mac	comp	x y z	ls
+Overtravel	24	0	mac	comp, cab	x y z	ls
-Overtravel	24	0	mac	comp, cab	x y z	ls
Overtemp.	contact	open	mac	cab	x y z s, tsr	tstat
Emergency Stop	contact	open	mac	cab	esr	es button
Position (6 ch)	5	5	mac	cab	x y z s	encdr
Drive Ready *	24	24	cab	comp	x y z s	driver
Overtemp.	24	0	cab	comp		tsr
Emergency Stop	24	0	cab	comp		esr
Machine Power	24	24	cab	comp		mcr
Position (6 ch)	15	15	cab	comp	x y z s	driver
Drive Enable	24	24	comp	cab	x y z s	mcc
Drive +EN	24	24	mac	cab	x y z	ls
Drive -EN	24	24	mac	cab	x y z	ls
Watchdog	contact	open *	comp	cab	esr	mcc
TCS (Torque)	10	<> 0	comp	cab	x y z s	mcc

Notes:

(1) *Active* watchdog state means a fault condition.

(2) The Drive Ready signal is termed Drive Fault in the MCC documentation.

The motion directions correspond to the polarity of the TCS signal and to the state of the +EN and -EN signals, as described in the following table:

Polarity of Motor Connections					
axis	+TCS	-TCS	+EN	-EN	plus (+) is
X	-	+	+	-	left
Y	-	+	+	-	outward
Z	+	-	-	+	up
S (velocity)	-	+	NA	NA	clockwise
S (torque)	+	-	NA	NA	clockwise

5. SYSTEM OPERATION

This section defines the coordinate systems and units of measurement, and describes the main operational modes and the startup procedure.

5.1. Coordinate Systems, Motion Directions and Units

The main coordinate system is the *machine system*, based on the directions of the main travel axes and spindle. The mechanical reference is established by the home limit switches and the encoders' marker to a very high repeatability (estimated 0.0005 inch). The homing procedure is described in a separate paragraph. The following directions of motion have been assigned to each axis:

Axes Motion Directions	
Axis	Positive Direction
X	left (towards motor)
Y	outward (towards motor)
Z	up
S	clockwise (looking down)

In addition, the system will support auxiliary coordinate systems, defined by a special operating command, with reference to the machine coordinate system. These auxiliary systems will typically be attached to the features of a part or a fixture, or may be the result of a transformation calculated from results of a probe measuring sequence. The transformatiom may be translational, rotational or a combination of both, and will be expressed in a matrix form.

Units of length will be either INCHes or MMs, selectable. Units of velocity will be IPM (inch per minute) or MMPM (mm per minute). Units of angular velocity will be RPM (revolution per minute). Units of acceleration will be IPM^2 (inch per minute-square) and RPM^2 (revolution per minute-square). Linear acceleration will sometimes be defined in G (gravity acceleration). The unit system, INCH or MM will be selected by an appropriate command. The default selection will be INCH.

Calibration factors should be assigned to each axis to transform the kinematics from INCH or MM to EDGE (encoder edges). (Motion direction is aligned by setting a parameter in the Encoder Mode Byte - described separately). The following example for the X axis explains:

x [EDGE] = X [INCH] * 8000 [EDGE/REV] / 0.2 [INCH/REV]

where:

x	position in encoder edges
X	position in inches
8000	encoder resolution
0.2	ball-screw pitch

For the spindle the relationship is:

x [EDGE] = X [REV] * 4000 [EDGE/REV]

The calibration factors have to be translated to HEX, with negative numbers expressed as Two's Complement, for the MCC to interpret them correctly. Additional detailed information on the calibration factors specific to each sub-system are brought in the *Software - Machine Applications* section.

5.2. Main Modes

5.2.1. User Interface

Alphanumeric

This interface provides the user with a direct access to the real-time processor through a serial line connected to a dumb terminal. The terminal can be opened also on a Sun window, by using the Kermit communications program. The user then types commands directly to the MOSAIC shell (Conch), and recieves messages on the console. The command syntax is described separately. This interface is most direct and flexible, and is used mainly for applications' development.

Graphic

This interface is running on top of the alphanumeric interface, and provides a subset of the most useful commands, accessible through graphic interaction with windows, menus and icons. The commands generated are piped through the fileserver running on the Sun, via the VMEbus, and onto the MOSAIC shell running on the 68020 real-time processor. The actual interface is described separately. Operations such as manual moves, tool path display, monitoring, and messages are done and presented in a very intuitive way. Additional windows can be opened concurrently for other applications, including an *alphanumeric* interface window. The graphic interface is used for running machining applications.

Interactive or Batch

Both the alphanumeric and graphic interfaces enable interactive or batch operations. In interactive mode the user types in the commands directly for execution. He can use any command from the MOSAIC shell set or the APT interpreter set. In batch mode the user can run batch programs from disk files, consisting of any combination of valid shell or APT commands, including nesting of batch files.

Apt or Noapt

When the apt mode is on, commands typed in are interpreted by the APT interpreter, running on the Sun, and piped through the fileserver, via the VMEbus, to the shell. In noapt mode the commands are sent directly to the shell, using the shell command syntax. These modes are available in all the previously described modes.

5.2.2. Motion

Homing

Homing (or Zero Return) is done automatically by the MCCs on each axis. The sequence consists of moving towards the home limit switch (positive direction in our case); upon triggering the switch, the motion reverses direction and proceeds until the encoder marker is encountered. The zero position is then recorded - typically as (0,0,0,0). The spindle, being a direct drive, does not need a limit switch, and the marker signal can be used directly.

Jogging

Jogging is moving the axes manually at a fast rate. The operater will select an axis or the spindle, define a direction, and invoke motion. Motion is invoked by pressing a predefined software key continuously, until the target is reached and the key is released.

Indexing

This mode refers to independent motion of an axis towards a target position. A profiled motion is used, in which the motion proceeds with predefined acceleration, ramp velocity, and deceleration. All the axes, including the spindle, can be both indexed or jogged.

Interpolation

In this mode a few axes move concurrently, in a coordinated accurate manner. We currently support linear interpolation of 4 axis (3 travel axes + spindle). Circular and spline interpolations are planned.

Absolute or Incremental

All motion commands support either absolute or incremental motion. Absolute position is with reference to the home position, or to the origin of an auxiliary coordinate system. Incremental position is with reference to the last (current) position.

Status Inquiry

Various inquiries on system and motion status are available. System inquiries include command execution timing, command help, etc. Motion inquiries include current position and velocity, current kinematic modal values, axis status, etc. The answers are displayed on the console.

Closed-Loop Control

This mode refers to closed-loop force and torque control. This option is not available yet.

5.2.3. Descrete Events

Enable or Disable

The motor drivers may be enabled or disabled. When enabled, motion commands can be executed by the motors. The MCC automatically disables the driver if the driver signals *drive fault*, or on predetermined error conditions, such as VMEbus reset or MCC's watchdog disabled.

Alarms Handling

The system handles a variety of alarm conditions. These include limit switches' overtravel on each travel axis, emergency stop, motor overtemperature, motion errors, communications errors, etc. Each condition is handled separately. Currently the conditions are tested by the MOSAIC shell 'tight loop'. In the future we will consider the use of interrupts.

5.2.4. Setup and Calibration

Initialization

The initialization procedure reads and executes the *mosaicrc* batch file. The procedure may alternatively be incorporated in the MOSAIC shell's executable for speed. During initialization, sub-systems' parameters sets are loaded over the VMEbus. Also, default units of measurement and coordinate system are selected.

Coordinate Systems

Any number of coordinate systems may be defined and stored on disk file. Each coordinate system is defined as a translational and rotational transformation from the mechanical home position. When a coordinate system is applied, all subsequent absolute moves are referenced to it.

Tuning

Gains and dynamic parameters may be tuned automatically or manually. These include the proportional, integral, velocity feedback and velocity feedforward gains, deadband compensation, maximum velocities and accelerations, and the following and velocity errors.

5.3. System Startup Procedure

The following section provides information on the current system startup procedure. This type of information is continuously changing, and should be applied with care. System directory structure is included in the *Software - Software Structure* section.

First, the power should be turned on on the electrical cabinet and the VME microrack. Then, the VMEbus should be reset manually. Verify that all the green and yellow LEDs are lit on the motion controllers, to indicate that there are no fault coditions like activated limit switches or other protection circuits. Verify also normal conditions on the front panel of the other processors.

Accessing the machine tool is currently performed in one of two ways. The most flexibile mode of access involves the use of the Sun as file storage device and the use of the additional dumb terminal with the serial line connection, or, the Sun in conjunction with Kermit for issuing commands. Alternatively the machine tool may be controlled through the graphical interface with a reduced instruction set. A description of the start-up procedure for both modes is given below.

The initial steps for access are the same in both modes. The user enters the */cadman.usr/src/local/mosaic/src/mosaic* directory on the Sun. Then the following commands are issued from any account with its root on *MOSAIC* if the graphical interface is not desired, or from the *psi* account with the user name *greenfel* if the graphical interface is desired:

vmeload

This readies the Pacific Micro CPU for down-loading the MOSAIC shell. At this point the system will respond with a message instructing the user to enter a starting address followed by a "y" informing the system that it may continue.

g f8000

This is the starting address on the CPU for the MOSAIC shell. This command is to be issued on either the dummy terminal or from a window running Kermit.

y This response is entered in the window used to issue *vmeload*. It informs the system that the CPU now knows where it should load the MOSAIC shell.

vpl mosaic
This command results in the down-loading of the MOSAIC shell (Conch) onto the Pacific Microsystems cpu. After a short while the system will ask if the user would like to use a fileserver. The fileserver allows the user to send batch files to the shell as well as to run the graphical interface.

y This is entered in response to the question for running the fileserver. It is issued on the terminal or window asking if a fileserver is desired.

At this point the set-up procedure differs depending on the mode of operation. To operate the machine tool without the graphical interface type:

vfileserver
This command starts running the fileserver on the Sun computer. It is used to handle file transfers between the Sun and the Pacific Micro CPU. It is often useful to issue this command with a "&" so that it runs in the background.

y Is entered on the terminal or window asking if *vfileserver* is running. It is crucial that this response is entered after the fileserver is running otherwise an error will occur.

At this point the user may issue commands controlling the machine tool, send programs to the machine tool and reconfigure the machine-tool's parameters.

To run the machine tool with the graphical interface the following procedure should be followed (note: the user must be logged as *psi*):

cd This return the user to *psi* home directory.

cd greenfel
Move to the *greenfel* sub directory.

x This starts up X-tools on the Sun computer. The display will blank and a bitmap of the machine tool will be displayed on the left half of the screen. A small window will appear in the lower right half of the screen. In this window the following commands should be issued.

set_up
This is a script file which invokes all of the remaining commands for using the graphical interface.

When the interface has been completely installed, i.e. the control panel is fully displayed on the screen, do the following:

y Is entered in respose to the question asking if *vfileserver* is running. This completes the start up procedure for running the machine tool with the graphical interface.

6. SOFTWARE

This section contains details of software requirements in the various levels of implementation. The current version of the document concentrates on the Motion Controllers' Interface, basic Control Applications and an extended APT Interpreter. Also, guidelines are provided for the User Interface and the Manufacturing Language.

6.1. Operating System

6.1.1. Real Time Operating System

The SAGE real time operating system has been adopted from robotic applications developed at NYU.

The implementation of SAGE to MOSAIC is to be defined. It involves the support of the VMEbus, fileserving interface, and multi-processing.

6.1.2. MOSAIC Shell (Conch)

The MOSAIC shell (conch) is the system's command interpreter running under SAGE on the real-time processor (the Pacific Microsystems 68020 CPU). Currently it is a single process; we are considering the benefits of multitasking, with or without interrupts and signals. A more suitable solution for the computationally intensive calculations, such as interpolations and inverse kinematics, will be to delegate those to additional real-time processors.

The shell handles the following groups of commands:

(1) Motion commands such as absolute and incremental moves, interpolated moves with various control schemes, spindle operation, setting of modal parameters (default acceleration, feedrate, etc.), homing, and motion status verification. These commands typically start with the letters *mc*. The large variety of motion primitives available enables a full motion control flexibility to be utilized by the other interpreters (e.g. APT).

(2) Communications with the motion controllers such as loading a command and reading status.

(3) System services such as reading batch files, path names, command help, display mode, timing of command execution, and query-waits.

(4) Setting modes of operation such as APT interpretation, axes enabling and disabling, motion controllers' parameters setting (e.g. gains).

(5) The shell is expandable, and will include in the future commands pertaining to other devices.

The current set of commands available on the MOSAIC shell follows:

? command

Print the name of all conch commands that begin with the string pointed to by 'command'.

apt

Begin taking input from the Apt interpreter on the host.

batch [batch_command_file]

Open a batch command file on the host. Batch files may be nested.

ebatch

Exit a batch command file.

bye

Exit conch.

connect

Connect to host fileserver.

disconnect

Disconnect from host fileserver.

exit

Exit conch.

help command

Print the name of all conch commands that begin with the string pointed to by 'command'.

init

Execute the conch's initializing batch command file.

logout

Exit conch.

mcaccel [accel(G)]

Set global maximum acceleration.

mccommand axis command_bytes

Manually send a command to a motor controller board.

mcdisable axis|all

Disable the feedback loop of a motor axis.

mcenable axis|all

Enable the feedback loop of a motor axis.

mcerror axis|all following|velocity

Print the following or velocity error of the specified axis.

mcfeed [rate(inches/min)]

Set or report global feed rate.

mchome axis|all

Perform homing sequence for a specified axis.

mcinit

Initialize the motor controllers.

mcparms axis|all

Set parameters for a motor axis.

mcpdelta x(in.)|, y(in.)|*, z(in.)|* [s(rev.)]*

Move travel axes (plus, optionally, the spindle) to given position increments using profiled move.

mcpercent [feed|spin] [factor(%)]

Report or change the scale factor for feed rate and/or spindle speed.

mcpmove x(in.)|, y(in.)|*, z(in.)|* [s(rev.)]*

Move travel axes (plus, optionally, the spindle) to a given position using profiled move.

mcposition [axis]

Print the current position of the specified axis.

mcrequest axis command_bytes

Manually send a request to a motor controller board.

mcrevision mpid

Print the software revision number of a motor controller board.

mcspin [rate(rpm)]

Run spindle at given rate (rpm), or report the current spindle speed.

mcstatus axis|all alarm|command|input|motion [motion|io|timer]

Print the specified status of an axis.

mcstop axis|all

Abort motion in an axis, with deceleration.

mctdelta x(in.)|, y(in.)|*, z(in.)|* [s(rev.)]*

Move travel axes (plus, optionally, the spindle) to given position increments using tracking move.

mctmove x(in.)|, y(in.)|*, z(in.)|* [s(rev.)]*

Move travel axes (plus, optionally, the spindle) to a given position using tracking move.

mctolerance axis|all following|inposition|velocity [edges|edges/sec]

Set or display following error, in-position, or velocity error tolerance.

mcwait axis|all home|less|greater|busy|locked|done|within [pos]

Wait for the specified status in an axis.

mcwithin [within(inches)]

Set or report global travel-motion target tolerance.

noapt

Turn off Apt interpretation.

nosilent

Turn off silent execution of batch files.

path [batch_pathname]

Set the default batch file pathname.

querywait

Wait until operator replies to prompt.

quit

Exit conch.

repeat count [command arg1 arg2 ...]

Performs the command indicated by its arguments and reports the time needed to execute it. It has two variants. As "repeat" it is given a count and the command is executed that many times. As "time" the commmand is performed only once.

silent

sleep [seconds]

time [command arg1 arg2 ...]

Performs the command indicated by its arguments and reports the time needed to execute it. It has two variants. As "repeat" it is given a count and the command is executed that many times. As "time" the commmand is performed only once.

6.1.3. Real Time Libraries

The HIC real time hierarchical control system has been adopted from robotic applications developed at NYU. Currently it is linked to the MOSAIC shell program, but is not actively used.

The implementation of HIC to MOSAIC is to be defined.

6.2. Machine Applications

This section contains the specification of the interfaces to each of the VMEbus application boards. It includes the board's communications with the real-time processor over the VMEbus, software parameter settings, and special provisions. Currently only the motion controllers' interface is specified.

6.2.1. Creonics Motion Controllers Interface

6.2.1.1. Communications

Communications with the Creonics Motion Controller Card (MCC) are described in the MCC manual, par. 4. The Read and Write procedures are implemented so that the complexity is transparent to the user. MCC alarm messages will be decoded and displayed on the operator's console. Alarms will typically generate a VMEbus interrupt (see MCC Message Register, par. 4.1.5 in the manual).

MOSAIC's particular implementation is to be defined.

6.2.1.2. Units Conversion Factors

The kinematic variables' conversion factors, from engineering units to encoder edges specific to the Creonics MCCs, are now defined (see also the section *System Operation - Coordinate System, Motion Directions and Units* for definition of units):

(1) Travel axes X, Y, Z:

8000 [EDGE/REV] / 0.2 [INCH/REV] = 40000 [EDGE/INCH]

position [EDGE] = position [INCH] * 40000 [EDGE/INCH]

velocity [EDGE/SEC] = velocity [INCH/MIN] * 40000 [EDGE/INCH] / 60 [SEC/MIN] * 65.536

acc-dec [EDGE/SEC^2] = acc-dec [INCH/MIN^2] * 40000 [EDGE/INCH]
/ 3600 [SEC^2/MIN^2] * 0.065536

also: 1 [G] = 1386000 [INCH/MIN^2]

(2) Spindle S:

position [EDGE] = position [REV] * 4000 [EDGE/REV]

velocity [EDGE/SEC] = velocity [REV/MIN] * 4000 [EDGE/REV] / 60 [SEC/MIN] * 65.536

acc-dec [EDGE/SEC^2] = acc-dec [REV/MIN^2] * 4000 [EDGE/REV]
/ 3600 [SEC^2/MIN^2] * 0.065536

The calibration factors have to be translated to HEX, with negative numbers expressed as Two's Complement, for the MCC to interpret them correctly.

6.2.1.3. Machine-Tool Parameters Setting

Refer to par. 5.1 in the MCC manual; the sections' numbers match. Use the unit conversion factors defined above to convert from engineering units to MCCs' units. Refer also to the *Creonics Motion Controllers Parameters Settings* appendix for the current values.

(1) Servo Gains

Will be set by using the Automatic Servo Setup Routines in par. 6.12 and 5.3.1 of the manual. Refer also to the *Calibration and Test Procedures* section of this document.

(2) Home Position

Will be defined by the HOME command described in the APT Interpreter section. Typically, but not necessarily, the home position value will be 0.

(3) Home and Overtravel Modes

Home and Overtravel Modes		
Parameter	Setting	Notes
Software Overtravel Mode	Non-Fatal (0)	
Hardware Overtravel Switches	Not Used (0)	general input
Software Overtravel Limits	Used (1)	Not Used (0) for spindle
Overtravel Switch Polarity	Normally Closed (1)	not relevant
Home Switch	Used (1)	Not Used (0) for spindle
Home Direction	Positive (0)	
Marker Pulse	Used (1)	
Home Switch Polarity	Normally Closed (1)	
Drive Fault Input	Used (1)	
Drive Fault Input Polarity	Normally Closed (1)	

(4) Maximum Travel Limits

Since we are using limit switches, there is no need (currently) for using software overtravel limits. Therefore, the values assigned here are the maximum (and minimum) allowed by the MCCs.

(5) Maximum Velocity, Acceleration and Decelation

Will be set by using the Automatic Servo Setup Routines in par. 6.12 and 5.3.2 of the manual. If these values are excessively high, set to the following nominal (safe) values:

Maximum Velocity, Accel. and Decel.			
Axes	Max. Velocity	Max. Accel.	Max. Decel.
X Y Z	200 [INCH/MIN]	1386000 [INCH/MIN^2]	1386000 [INCH/MIN^2]
S	2000 [REV/MIN]	6930000 [REV/MIN^2]	6930000 [REV/MIN^2]

Note: 1386000 [INCH/MIN^2] = 6930000 [MOTOR_REV/MIN^2] = 1 [G]

where the bell-screw pitch is 0.2 [INCH/REV]

(6) Following Error Tolerance

Will be set to the maximum value allowed by the MCCs, to avoid irrelevant errors during acceleration and deceleration.

(7) Deadband Compensation

Will be set by using the Automatic Servo Setup Routines in par. 6.12 and 5.3.1 of the manual. If too high, set manually to a low value (between 0 and 5).

(8) Feedforward Gain

Will be set by using the Automatic Servo Setup Routines in par. 6.12 and 5.3.3 of the manual.

(9) Encoder Mode

Voltage and Encoder Polarities will be set to:

Encoder Voltage Polarity		
Axis	Voltage Polarity	Encoder Polarity
X Y	Inverted (1)	Inverted (1)
Z S	Normal (0) (Inverted in velocity mode)	Normal (0)

Encoder Decode Mode Select will be set to Mode 4: 4X Quadrature Decode (100).

(10) Axis/Drive Type Select

Will be set to: X Y Z S - Linear with Analog Current Loop (00)

Note: The spindle is also defined as a linear axis to enable absolute positioning.

(11) In-Position Tolerance

Will be set to +-7 encoder edges.

(12) Velocity Error Tolerance

Will be set to a large value, to avoid nuisance errors.

(12) Unwind Constant

Currently not needed since the spindle is defined as a linear axis rather than rotary. Set to 4000 [EDGES PER AXIS REVOLUTION].

6.2.1.4. Dexman Parameters Setting

To be defined.

6.2.2. Image Processor Interface

To be defined.

6.2.3. General Input/Output Interface

To be defined.

6.3. Control Applications

This section deals with motion control, from the single-axis control, to linear, circular and spline interpolations (multi-axis coordinated motion). It also deals with control of the analog and descrete devices and sensors, like programmable fixtures, force sensors, vision, etc. Currently, only basic moves and main alarms are supported.

6.3.1. Motion

6.3.1.1. Machine Axes and Spindle

This section defines the basic move commands and related modals.

Regular Move:

The standard move uses *profiled velocity* with prespecified acceleration, ramp velocity and deceleration. Both absolute and incremental moves are supported. The move velocity (feedrate), acceleration and deceleration are defined as a modal by a MOSAIC shell command. In addition to the regular feedrate setting, a *percent* mode enables to change the scale factor of a feedrate or spindle jog speed. The default settings are:

default_feedrate = 0.1 [INCH/MIN]

default_feedrate_scale_factor = 100%

default_accel_decel = 0.1 [G]

For interpolated moves, all the kinematic modal values are defined as. pathwise. Currently, linear interpolations are done by precomputing the accelerations and velocities for all the axes, and commanding each axis for an independent motion (*Track* facilities will be implemented also). Each axis is assigned values as follows:

axis_speed = axis_travel / path_travel * feedrate

axis_accel_decel = axis_travel / path_travel * accel_decel

Where:

path_travel = +SQRT (X^2 + Y^2 + Z^2)

Spindle Jogging and Positioning:

The spindle has two modes of operation: *jogging* and *positioning*. The jogging provides constant velocity with specified acceleration and deceleration. When a position value is specified for the spindle in a move command (see syntax in the MOSAIC shell and APT interpreter), the spindle positioning mode is automatically selected. Spindle homing is done automatically when this switch between modes occurs. The jogging mode returns automatically upon the issuing of a spindle spin command.

In the positioning mode ,the spindle is assigned velocity and acceleration values according to the following two cases:

(1) Spindle moves independently:

spindle_speed =.feedrate * scale_factor

spindle_accel_decel = accel_decel * scale_factor

Where:

feedrate and accel_decel values are the global modals

scale_factor = 40000 [EDGE/INCH] / 4000 [EDGE/REV] = 10 [REV/INCH]

Note: The spindle rate in this case is determined so that it will take the same time for the spindle and axis motors to perform one revolution, when they use the same global feedrate modal.

(2) Spindle moves in coordination with travel axes:

spindle_speed = spindle_travel / path_travel * feedrate

spindle_accel_decel = spindle_travel / path_travel * accel_decel

Where:

feedrate and accel_decel values are the global modals

path_travel = +SQRT (X^2 + Y^2 + Z^2)

Motion Conditions:

Starting the execution of the next command will always depend on the current status of the axes. Usually, the next command will have to wait until a motion condition, pertaining to the previous command, has been met. The Creonics MCCs provide the following optional conditions (refer to par. 5.1.14 in the MCC's manual):

- Home complete.
- Position less than absolute position.
- Position greater than absolute position.
- Axis busy.
- Axis locked.
- Axis done

All these options are supported by the MOSAIC shell (described separately). They will be automatically assigned before or after each command by the motion interpreters (APT and AML), as applicable. Typically motion commands will be subject to the *position less/greater than absolute position* condition, meaning that the new command will wait until the axes have reached their targets within a prespecified tolerance, called the *within tolerance*. This enables smoothening of motion between commands. The tolerance is defined as a modal by a MOSAIC shell command. The default setting is:

default_within_tolerance = 0.0005 [INCH]

The *position less/greater than absolute position* condition is implemented as follows:

Motion Wait Condition		
SIGN (Xt - Xc)	Condition Type	Xw
>0	wait for greater	Xt - T
<0	wait for less	Xt + T
=0	ignore	

Where:

'w' refers to the within position
't' refers to the target position
'c' refers to the current position
All positions are absolute

This check is done for each axis separately, and the wait condition ends only after ALL axes are inside the within tolerance. Excluded is the spindle, when it is in the *jog* mode.

Speed and acceleration setting commands will have the option to be executed without waiting for the axes to reach their targets, thus enabling the change of feedrates, spindle speeds and accelerations on-the-fly. E.g. this is needed for closed-loop force control.

6.3.1.2. Dexman

To be defined.

6.3.2. Sensors

To be defined.

6.3.3. Descrete and Analog Devices

6.3.3.1. Main Alarms

General alarm conditions are currently handled by the spare input channels on the spindle section of the MCC, as follows:

General Alarms Interpretation		
Input Channel	Mode	Action
Home	general input	overtemperature - stop all axes.
+Overtravel	general input	emergency stop message only
-Overtravel	general input	machine power on message only
Drive Fault	general input	stop all axes

Machine motion alarms will be interpreted by the MCC as follows:

Machine Motion Alarms Interpretation		
Input	Mode	Action
Home	Home	MCC homing sequence on the axis involved
+Overtravel	general input	stop all axes, excluding spindle
-Overtravel	general input	stop all axes, excluding spindle
Drive Fault	general input	stop all axes, excluding spindle

6.4. Planning Applications

Currently available are off-line tool-path generation on the Anvil-5000 cadcam system, and automatic machining setup for prismatic parts on the Machinist expert system. See applicable documents.

More to be defined on the integration of these systems with the MOSAIC system.

6.5. Manufacturing Languages

6.5.1. APT Interpreter

This section contains an APT interpreter to enable the operation of the machine tool in the full machining capabilities. The interpreter supports conventional APT commands, as well as a set of additional APT-like commands pertaining to machine operations. Some APT commands have been extended to encompass more functionality. The commands are of three categories:

(1) Major APT commands, mainly concerning the part definition, like GOTO, SURFACE and GODLTA.

(2) Post Processor (PP) commands, mainly concerning machine specific parameters, like FEDRAT, LOADTL and SPINDL.

(3) Machine operation (MO) commands and special commands, mainly operations not concerning machining directly, like INIT, HOME and JOG, and motion commands not supported by APT.

The post processor commands are not a part of the APT system, but are typically supported by the associated post processors. The machine operation commands are also not a part of the APT system, and are unique to our installation, though they are of a generic nature for machine control. They are intended to form the basis for a flexible operating environment for the machine, and to provide a convenient interface for the implementation of the manufacturing language and the user interface.

The APT command syntax is:

MAJOR_WORD/MINOR_WORD, ..., value, ..., ...

The major and minor words are up to 6 characters in capital letters. In the list below small letters indicate user supplied values. the slash and the commas are required. Spaces are allowed. Some of the commands are not supported to their full functionality specified in the original APT specifications, and some have additional functionality.

The following list is sorted to two groups:

- Machining - mainly Apt and post-processor commands.
- Operations - mainly machine operation commands.

6.5.1.1. Machining

FROM/x, y, z [, s]

Go at maximum axes speed to the desired location in the current coordinate system. This will be the starting point for a tool path. If not programmed, the value of (0,0,0,0) will be inserted at the begining of a tool path. If the s value is not specified, then the spindle is not servo-positioned, and may be jogged for cutting. If the previous spindle mode was jogging, the mode is automatically switched to positioning, and homing sequence is executed.

GOTO/x, y, z [, s]

Go at a straight line (linear interpolation) to the desired location in the current coordinate system, at the current feed rate. If the s value is not specified, then the spindle is not servo-positioned, and may be jogged for cutting. If the previous spindle mode was jogging, the mode is automatically switched to positioning, and homing sequence is executed.

GODLTA/dx, dy, dz [, ds]

Go incrementally at a straight line (linear interpolation) to the desired incremented location in the current coordinate system. If the ds value is not specified, then the spindle is not servo-positioned, and may be jogged for cutting. If the previous spindle mode was jogging, the mode is automatically switched to positioning, and homing sequence is executed.

FEDRAT/[NOW,] feed_rate [, IPM, IPR, MMPM, MMPR]

Sets the modal feed rate value as feed_rate, in the current unit system. If MMPR (MM per revolution) or IPR (INCH per revoluton) are specified, the feed rate will be a function of the spindle revolutions (this option is used for tapping). The default setting is IPM. If the value exceeds the maximum allowed value, then the maximum value will be inserted, and an operator's message issued. When the NOW modifier is specified, the feedrate value is assigned immediately to the current move, regardless of any 'wait for condition' settings.

```
OFSTNO/coordinate_system_name
```

Sets the modal coordinate system defined as coordinate_system_name. The name is defined by the following assignment:

```
coordinate_system_name = MATRIX/  q11, q12, q13, c1,
                                  q21, q22, q23, c2,
                                  q31, q32, q33, c3
```

where the elements q[i,j] are those of the rotational transformation and the elements c[i] are those of the translational transformation.

```
SPINDL/RPM, spindle_speed [, CLW, CCLW]
      /OFF
```

Turns the spindle on or off. The spindle speed is set to spindle_speed at the direction specified by CLW (clockwise) or CCLW (counter- clockwise). CLW is the default. If the value exceeds the maximum allowed value, then the maximum value will be inserted, and an operator's message issued. If the previous spindle mode was positioning, the mode is automatically switched to jogging.

```
COOLNT/ON
      /OFF
```

Turns the coolant on or off.

```
LOADTL/tool_number [, OSETNO, offset_number] [, LENGTH, length] [, PROBE]
```

Loads tool number tool_number and assigns a tool offset and a length offset. This function will be detailed when the tool loading facility will be installed.

```
CYCLE/DRILL, FEDTO, dz, RAPTO, rl, RTRCTO, r2 [, IPM, IPR], f
      /TAP             DWELL, d
      /BORE
      /PECK            INCR, i
      /ON
      /OFF
```

This command invokes a canned cycle operation of one of the listed choices. Refer to details in the Fanuc 10M operator's manual and the Synermation post processor manual.

```
SURFACE/x_center, y_center, z_center, x_vector, y_vector, z_vector, radius
```

Moves in a circle of the specified radius, the center of which is at the specified center. The circle lies in the plane defined by the direction vector specified. The direction of the cut is defined by the sign of the vector. E.g. the direction vector (0,0,-1) is the XY plane and the cut direction is clockwise. The previous GOTO is assumed to be the start point of the circle, and the next GOTO is assumed to be the end.

```
ARCSLP/START, s, ENDARC, e, RADIUS, r [, CLW, CCLW]
```

Moves in a circle in the direction determined by CLW or CCLW. CLW is the default. The circle center point is defined by the start angle s and the end angle e and the radius r. Used for circles in the XY plane only.

SPLINE/ON
/OFF

Sets the spline modal on or off. When the spline mode is on, the GOTO points are treated as points on a bicubic spline. The mathematical relationships are to be defined..

DISPLAY/message

Display a message on the operator's console.

REMARK/string

A comment inside a program. To be ignored during execution.

DELAY/delay
/REV, number_of_revolutions

The system will delay for the amount of time (in seconds) specified. If REV is specified, the delay will be for the number of spindle revolutions elapsed.

END

End of program. Stops all motion, and resets all defaults.

STOP

Stops all motion on travel axes (excluding the spindle in jog mode) Motion may be restarted by the appropriate start command.

OPSTOP

Stops travel axes (excluding the spindle in jog mode) motion if the optional stop modal was previously set. Motion may be restarted by the appropriate start command.

CHECK/XSMALL, xs, XLARGE, xl, ...

Checks that all motion commands are within the specified space, defined in the machine coordinate system.

6.5.1.2. Operations

INIT/ENABLE [, X, Y, Z, S] [, PARAMS]
/DISABL [, axes_list]

Enables or disables the specified drives and verifies that there is no drive fault, and that there are no motion alarms or other alarms. If the modifier PARAMS is included, loads parameters to specified axes. Status message will be issued to the operator. The command without the axes list will refer to all axes.

TUNEUP/[GAINS, DYNMIC, FEDFOR] (cancelled for safety and configuration control)

Invokes the Automatic Setup commands described in par. 5.3 in the MCC manual. The tuneup cycles are gains, dynamics and feedforward. If no modifier is specified, then all tuneup cycles are executed. The results are stored in a setup file that is to be read into the MCC whenever power is applied. Refer for more information in the Modes of

Operation paragraph, and in the MCC manual, par. 5.3 and 6.12.1.

HOME/[axis_name, position_value ......]

Homes all the specified axes and assigns the position value specified as the home coodinates in the machine coordinate system. If no modifier is specified, all the axes are homed, and the position values of 0 are assigned. Refer for more information in the Modes of Operation paragraph, and in the MCC manual.

JOG/x, y, z [, s]

Moves at maximum speed to the specified location in the current coordinate system.

UNITS/[INCHES,MM]

Selects the unit system. INCHES is the default.

START

Starts a program run, or restarts a stopped motion.

MODE/ON [,OPSTOP, DRYRUN, SGLBLK, other_modes]
/OFF [, mode_list]

Sets on or off all the modes listed. OPSTOP is an optional stop within a program. DRYRUN is a fast replay of a program without cutting. SGLBLK enables to run a program one command at a time by pressing STARTs.

HOLD/[SPINDL, FEED]

Stops motion of the spindle and/or the travel axes until a restart command. If no modifier is specified, all motion is stopped.

EMS

Stops motion of the spindle and the travel axes and then disables all drivers.

6.5.2. AML Interpreter

An interpreter for robotic operations, intended for use with Dexman in conjunction with the APT machine interpreter, is to be defined. The language to be chosen will depend on the final selection of the manipulator's geometry. The three main contenders are AML, VAL and APT, or a mixed syntax thereof.

6.5.3. Advanced Manufacturing Language

A universal manufacturing programming tool is needed to support in real-time the multi-device MOSAIC environment. While many manufacturing languages exist, they are typically designed with a specific application in mind. Among them are the Automatically Programmed Tools (APT) used for machining processes; IBM's Manufacturing Language (AML and AML/X) for robotics, machine vision and computer aided design; the Cell Manufacturing Language (CML) which interprets existing languages relating to various applications; and Relay Ladder Logic used for programming descrete operation devices.

Machine	Tool	Process	Pre-Process Object		Post-Process Object
Nominative	Ablative	Verb 1	Accusative 1	Verb 2	Accusative 2
The CAD system	uses cad algorithms	to process	data	to make	a tool path.
The expert system	uses rules and databases	to process	knowledge	to make	a setup plan.
The loading robot	uses a gripper	to position	cutting tools	to create	a tool-set in the carousel.
The loading robot	uses a gripper	to position	the vise	to create	the generic set-up.
The loading robot	uses a gripper	to position	the stock in the vise	to create	the first set-up.
The vise	uses automatic jaws	to clamp	the stock	to create	a fixed setup.
The milling machine	uses tool 1	to cut CNC routine 1	into the block	to make	feature 1.
The milling machine	uses tool *n*	to cut CNC routine *n*	into the block	to make	feature *n*.
The dextrous manipulator	uses a gripper	to reposition	the stock	to create	new setups *n* to *n* + 1
The vision system	uses algorithm 1 and database 1	to analyze	the vise area	to verify	a chip-free setup environment.
The vision system	uses algorithm 2 and database 2	to analyze	the flank of the tool	to verify	tool conditions.
The piezoelectric dynamometer	uses algorithm 3 and database 3	to analyze	tool forces	to verify	tool conditions.

Figure 6.1. The Manufacturing Language User-Level Structure

A real-time, object-oriented manufacturing language is proposed to program the various applications under a unified programming environment. This language will also serve as a universal medium for manufacturing data, and as a powerful educational tool. In the description of the concept of the language, we put the emphasis on the abstract level of the concept, rather than details of language structure and syntax. The interaction with cadcam systems is stressed, as well as with expert systems, such as Machinist which provides some of the human machinist's skills in process planning.

Figure 6.1 proposes a syntactical structure for this manufacturing language. The goal is to provide an abstraction of any generic manufacturing process in such a way as to be a basis for the universal manufacturing language. As an example, a generic machine tool holds a tool of some geometric description which is then set into motion (at a predescribed speed, feed and depth-of-cut) to cut a raw block. A combination of such cuts leads to the formed part. Other common processes are described in the same way.

In the most abstracted sense, the top line of the table shows that a *machine*, by way of a *tool*, applies a *process* to a *pre-process object* to *make* a *post-process object*. In language structure, these terms correspond to (in the second line of the table) the nominative, ablative, verb 1, accusative 1, verb 2 and accusative 2.

Each device can be treated in the same way as the machine tool. However a 'device' can also be viewed more abstractly. Hence the expert system is a device that uses rules and databases to process knowledge to make a setup plan. Going down the table, a somewhat chronological series of events is described involving the various functions of the loading robot, the vise, the milling machine and the dextrous manipulator. Note that the vision system and other sensors can be treated similarly.

The table and its extensions is the structure for the MOSAIC programmer. The programmer can use the *names (objects)* in the table and arrange them in sentences to program *actions*. Sentence construction is done by 'clicking' on named devices in the user console and specifying their actions from 'pull-down' menus. Timing and program flow-structure provisions, needed for real-time operation and for decision making, should be incorporated by extending the syntactical structure to support logic.

Existing Object-Oriented Programming languages are eligible as candidates for the implementation of the manufacturing language. Some of the more important ones are Smalltalk-80 , ADA and C++.

6.6. User Interface

The MOSAIC controller is configured to have an extensive graphic interface. Figures 6.2, 6.3 and 6.4 present the framework for this interface. This presentation has two goals in mind:

- to lay out a generic specification for the future configuration of MOSAIC, by way of describing its functionality.
- to provide guidelines for the graphic 'look' of the operator's console and the style of the user interaction.

The user interacts with the system through the Sun's terminal, rather than through a special operator's panel. He may open a window for each application. He uses the mouse to select options by clicking on icons, menus and dialogue boxes, and to move features by click-and-drag. Thus, the conventional operator's panel is emulated by the software in a more powerful and flexible version. Three main user-interaction screens are shown:

- The Designer/Planner Screen - figure 6.2.
- The Machine Operator Screen - figures 6.3.
- The Plant Operator Screen - figure 6.4.

Each of these screens represents a phase in the design and production cycle, and all the applications share common data bases of parts, tools and stock and a common user interface. Some of the functions described, such as cadcam, expert systems, process planning, alarm diagnostics, etc., already exist as stand-alone software packages available from various vendors. The view depicted here concentrates on the integration of these and new functions into a comprehensive design and operation environment. A description of each user screen follows.

6.6.1. Designer/Planner

The *Designer/Planner Screen* includes the following windows:

- *Design Window*:

This window is built around an object-based solid modelling cadcam system. The solid primitives relate to physical entities, like hole, wedge, pocket, etc., rather than to geometrical units (cylinder, box, etc.). This enables the system to use the part's model for the generation of manufacturing processes, such as the automatic creation of tool paths and a process sheet, with the aid of the process planning function described later. A variety of libraries is available for the designer, including parts, tools, and processes. Tools and fixtures may be overlaid on the part's design, and manufacturing process issues considered. These libraries may be attainable from local or factory-wide databases, as well as from nationwide databases. The easy accessibility to data and the automatic features of the system make it possible for the designer and process planner to be the same person, thus achieving higher efficiency.

- *Process Planning Window*:

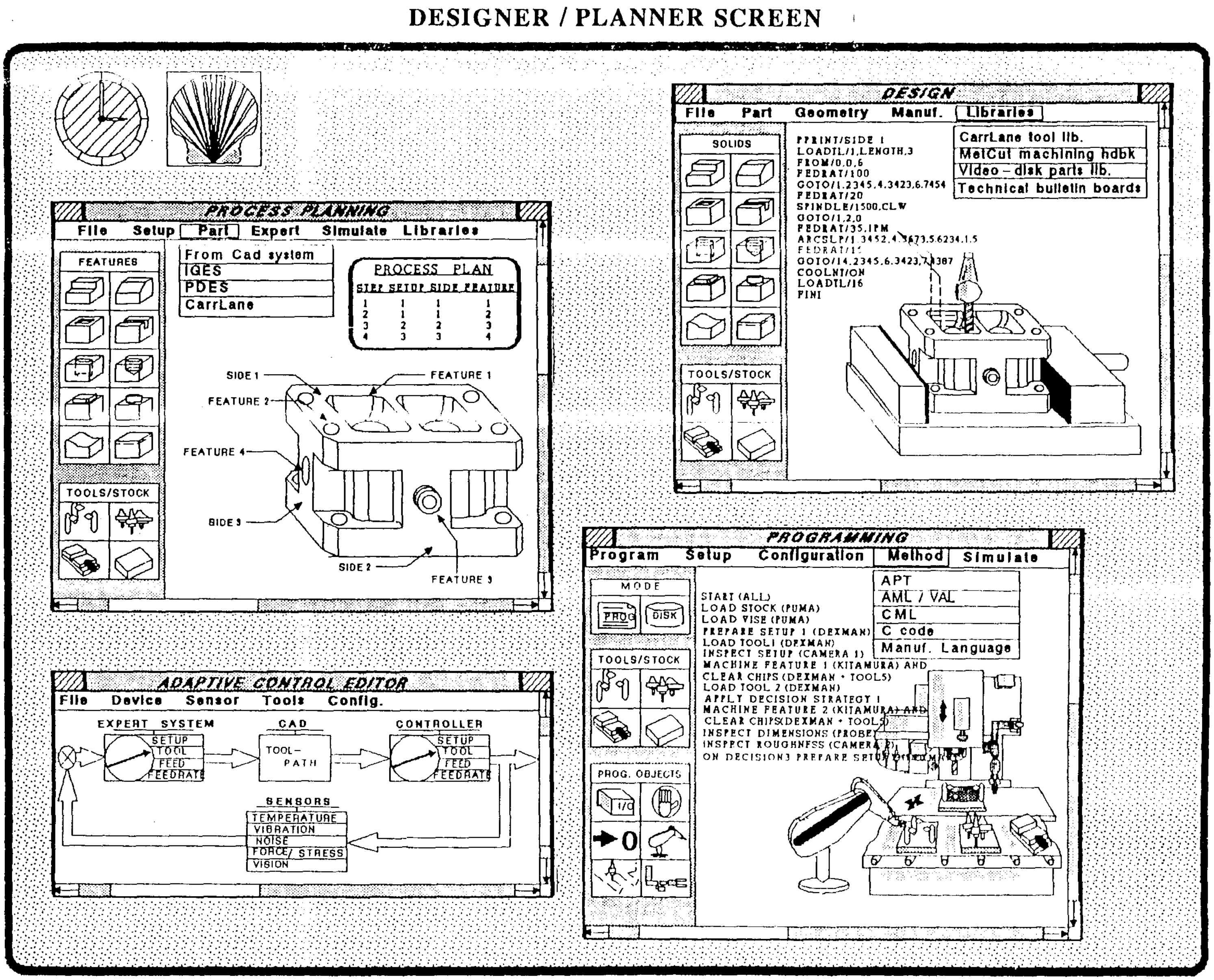
This window provides an automatic process planning capability. The part design is retrieved from a cadcam system, preferably in a universal part representation format like IGES or PDES. The part features' definition and sides are extracted from the object-based solid modeller automatically, and fed into a manufacturing expert system, in this case a machining expert. The expert system prepares the process plan, including the definition of setups and the order of operations, using inherent rules. As in the design window, the same libraries are available, and tools and fixtures may be called into the design. The final process may be simulated for verification.

- *Adaptive Control Editor Window*:

This window provides a graphic editor which enables the user to define the system operational rules (or control laws), as a function of the state of the system. Physical parameters such as tool temperature, vibration, noise and torque, and input from vision systems may be used to monitor the process. An expert system analyzes the data and makes decisions relating to the process. Typical decisions may be to change feeds or gains, change tools, modify a tool path, operation abort, etc. These decisions are fed to the cadcam system for regeneration of tool paths. The editor provides tools for the graphic representation of the full decision system, by using icons representing objects like gauges, parts, tools, tool paths, and decision switches.

- *Programming Window*:

This window provides a full scale programming capability , using a unified manufacturing language. The virtual setup of the machine is created by selecting menu-objects like manipulators, cameras, fixtures, etc., and placing them in the layout. Previously used layouts may be retrieved from library. Operations objects are also represented as icons like push-buttons, devices, etc. Most of the programming is done in a virtual teach-mode, whereby the



PROCESS PLAN

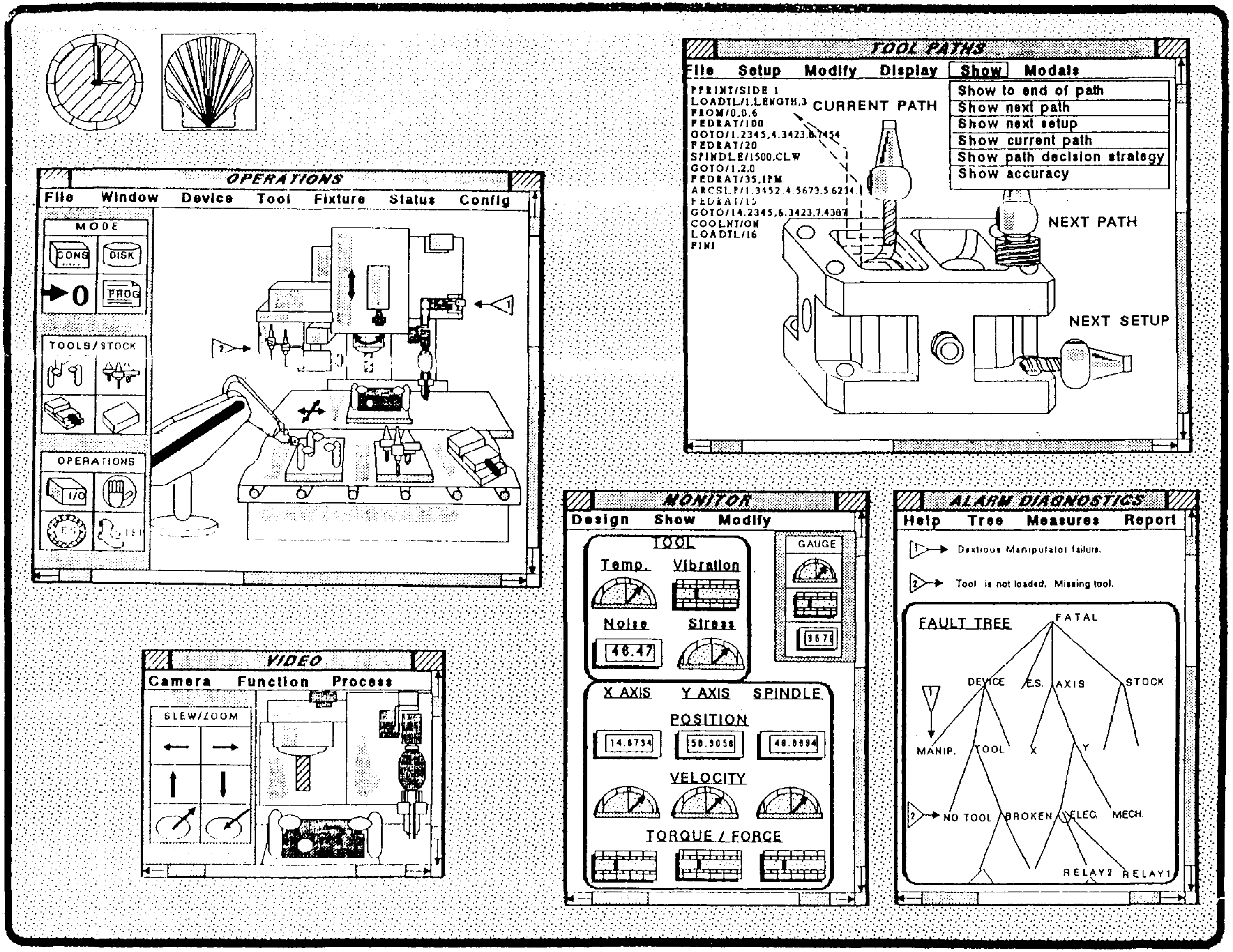
STEP	SETUP	SIDE	FEATURE
1	1	1	1
2	1	1	2
3	2	2	3
4	3	3	4

```
PPRINT/SIDE 1
LOADTL/1,LENGTH,3
FROM/0,0,6
FEDRAT/100
GOTO/1.2345,4.3423,6.7454
FEDRAT/20
SPINDLE/1500,CLW
GOTO/1,2,0
FEDRAT/35,IPM
ARCSLP/1.3452,4.3673,5.6234,1.5
FEDRAT/15
GOTO/14.2345,6.3423,7.4387
COOLNT/ON
LOADTL/16
FINI
```

```
START (ALL)
LOAD STOCK (PUMA)
LOAD VISE (PUMA)
PREPARE SETUP 1 (DEXMAN)
LOAD TOOL1 (DEXMAN)
INSPECT SETUP (CAMERA 1)
MACHINE FEATURE 1 (KITAMURA) AND
CLEAR CHIPS (DEXMAN + TOOL5)
LOAD TOOL 2 (DEXMAN)
APPLY DECISION STRATEGY 1
MACHINE FEATURE 2 (KITAMURA) AND
CLEAR CHIPS(DEXMAN + TOOL5)
INSPECT DIMENSIONS (PROBE)
INSPECT ROUGHNESS (CAMERA 2)
ON DECISION3 PREPARE SETUP 2(DEXMAN)
```

Figure 6.2. The User Interface - The Designer/Planner Screen

MACHINE OPERATOR SCREEN



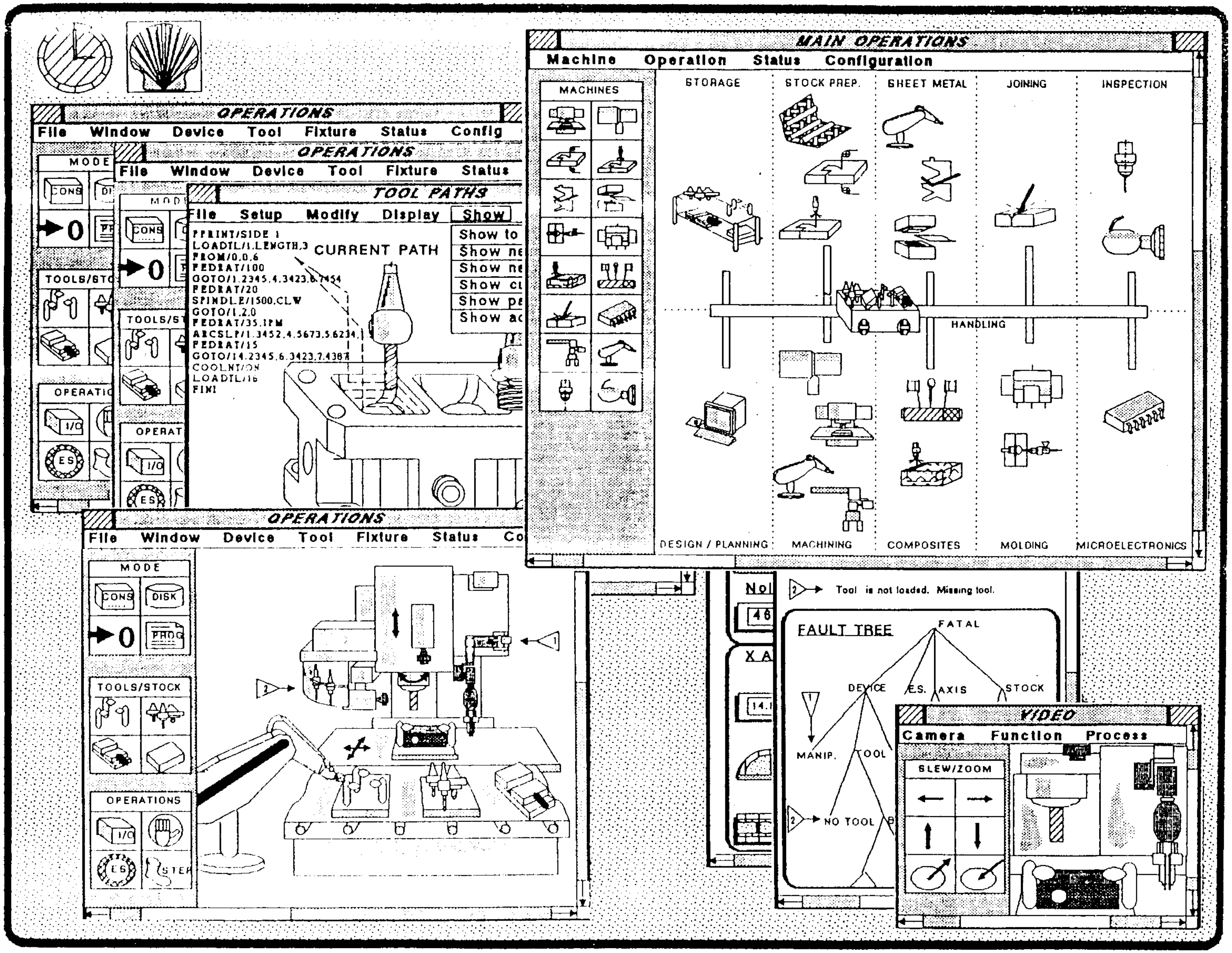
```
PPRINT/SIDE 1
LOADTL/1,LENGTH,3
FROM/0,0,6
FEDRAT/100
GOTO/1.2345,4.3423,6.7454
FEDRAT/20
SPINDLE/1500,CLW
GOTO/1,2,0
FEDRAT/35,IPM
ARCSLP/1.3452,4.5673,5.6234
FEDRAT/15
GOTO/14.2345,6.3423,7.4387
COOLNT/ON
LOADTL/16
FINI
```

1 Dextrous Manipulator failure.

2 Tool is not loaded. Missing tool.

Figure 6.3. The User Interface - The Machine Operator Screen

PLANT OPERATOR SCREEN



```
PPRINT/SIDE 1
LOADTL/1,LENGTH,3
FROM/0,0,6
FEDRAT/100
GOTO/1.2345,4.3423,6.7454
FEDRAT/20
SPINDLE/1500,CLW
GOTO/1,2,0
FEDRAT/35,IPM
ARCSLP/1.3452,4.5673,5.6234,
FEDRAT/15
GOTO/14.2345,6.3423,7.4387
COOLNT/ON
LOADTL/16
FINI
```

Tool is not loaded. Missing tool.

Figure 6.4. The User Interface - The Plant Operator Screen

programmer moves around the machine axes, manipulator gripper, parts, stock and other devices, by click-and-drag mouse operations, and selects canned operations, such as loading, cutting and inspecting, from menus. A program script is then created automatically, in any of a given selection of manufacturing languages. A fully concurrent operation of all components, though, can be achieved by a unified language only. Decision strategies, supported by the expert system, are incorporated into the program as modal statements. The final program may be simulated for verification.

6.6.2. Machine Operator

The *Machine Operator Screen* includes the following windows:

- *Operations Window*:

This window provides the machine operator with a full access to all the components of the manufacturing environment. The machine setup is created by moving and manipulating fixtures, stock, cameras, etc., according to the setup plan already defined in the programming window. This operation, naturally, demands high modularity of the machine, and the availability of all the subsidiary devices and tools over a transfer line such as a conveyor belt. Cutting tools do not have to reside in a local tool magazine, but rather called up whenever needed. The operator can move and manipulate objects in a similar way to the programming window, and, in addition, he can watch the changes on the screen in real time. He can run programs and watch their progress, or operate the machine in a manual mode, by selecting options from the menus or by click-and-drag. Alarm messages are displayed as flags that point at the component at fault.

- *Tool Paths Window*:

This window shows the current progress of a tool path, along with the cutting program prepared in the programming window. The operator can verify in real time the accuracy of the current path. He may also preview on-coming tool paths or part setups, and the path decision strategy if applied.

- *Monitor Window*:

This window provides the user with a flexible monitoring interface. Any combination of gauges, in any available meter style, may be constructed by the operator and possibly saved for future use. Each gauge is measuring a particular manufacturing parameter, such as axes' position and velocity, or tool's force, torque, temperature, etc.

- *Alarm Diagnostics Window*:

This window provides an alarm diagnostics facility, using a preconstructed fault tree, and expert system techniques. Cross reference of alarm flags with those presented on the operations screen is available. A few such windows may be opened simultaneously.

- *Video Window*:

This window provides a video monitor, opened as a window tool on the computer. It allows the operator to view the manufacturing scene remotely, by switching between cameras, and slewing and zooming them towards a target. A few such windows may be opened simultaneously.

6.6.3. Plant Operator

The *Plant Operator Screen* includes the following windows:

- *Main Operations Window*:

This window provides the plant operator with a layout of the factory floor. He can create or modify the layout, according to the physical layout, by selecting options from menus. He can move and manipulate the flow of stock and parts. He can zoom-in on a particular machine for a follow-up on the progress of the process, or for direct control in case of need.

- *Other Operations Windows*:

Any combination of machine-specific windows, such as those described before, for any machine under supervision of the plant operator, may be presented on the screen for his use.

6.6.4. Implementation

A number of new graphic interfaces featuring a user-friendly interface, using widows, icons and menus, have been introduced recently. The main two are the Open Software Foundation's MOTIF based on X-Windows developed at MIT, and Sun's NeWS. It is not clear yet which is going to evolve into an industry standard, and thus any selection is risky. Therefore, the graphic interface should be designed with portability in mind.

Currently the X-Windows environment has been chosen for demonstration purposes. This preliminary interface, shown in figure 6.5, features a small set of software buttons, through which the machine's axes and spindle can be manually moved with different speeds and directions. These functions will be improved to support click-and-drag style of axis moving. Position, velocity and torque meters will be displayed, as well as the planned and current tool path.

6.7. Software Structure

The following section provides information on the current system file-directory structure. This type of information is continuously changing, and should be applied with care.

/cadman.usr/src/local/mosaic

The effective root of the MOSAIC system. It contains three major sub-directories: *cmd*, *src.devel* and *src*. All additional directories listed below should have this path preceeding their name.

/cmd Contains the system parameters, the *mosaicrc* initialization file, and utility programs as well as other programs. When a request is made through thc *batch* command this directory is consulted to find the requested file.

/src.dev

Contains software under development for MOSAIC. It is broken into several subdirectories named: *apt, mosaic, hic, conch, h, util, motor, control* and *procopy* which contain software related to their names. For example *apt* contains software under development for the APT interperter.

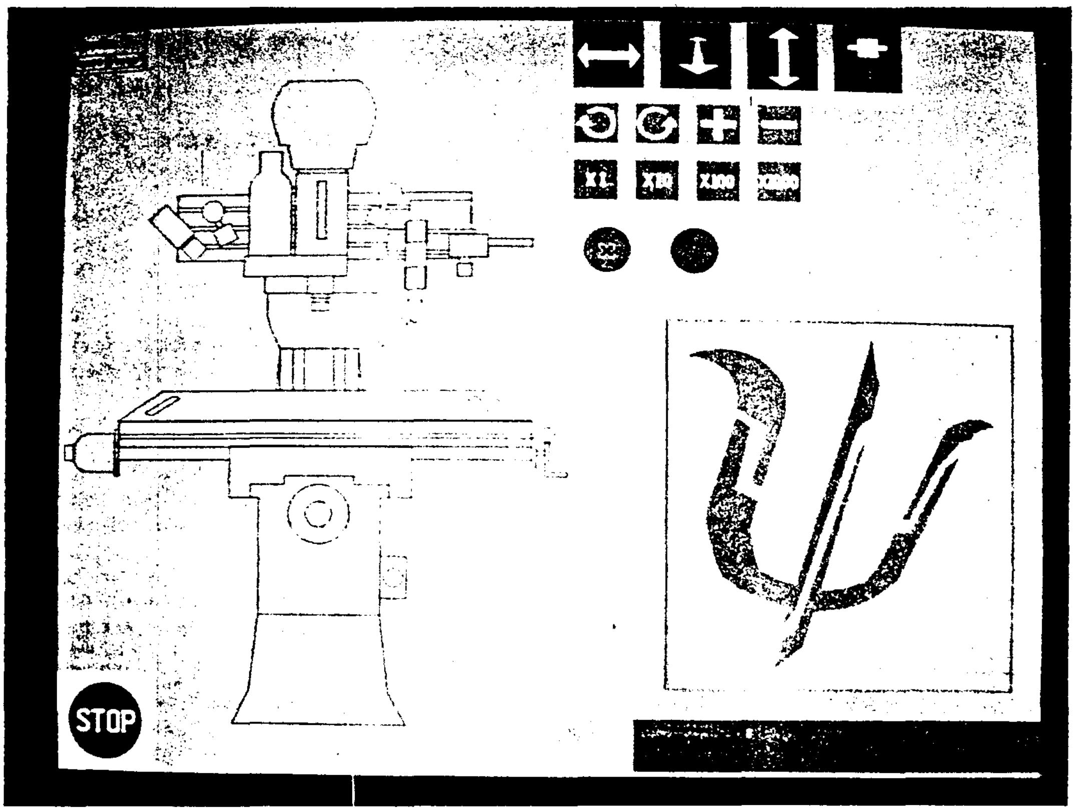


Figure 6.5. The Current Machine Operator Screen Implementation

/src Contains the source code for the version of the system currently implemented. It too is divided into several sub-directories: *apt, mosaic, hic, conch, h, util, motor* and *control.*

In addition to these directories a note should be made of the location of sage, the real time operating system implemented in the MOSAIC project. It is located at */robeye.d/local.*

7. CALIBRATION AND TEST PROCEDURES

7.1. Tuning and Calibration

Tuning of the motion control digital loops will be done on the motion controllers, according to the procedures described in the motion controller manual, *Automatic Servo Setup Routines*, par. 6.12 and 5.3.1. The *cmd* directory contains a few sample batch files used for tuning, typically named **tune**. After running these procedures the resulting gains should be inserted manually into the parameters' files. Although this could be done automatically, it is not advisable for safety and configuration control reasons. A few words of advice and caution, based on our current experience, should be noted:

- Sometimes the builtin automatic tuning routines respond in an unexpected vigorous way, in particular the *Tune Dynamics* routines. Precaution should be taken by keeping the emergency stop button handy.
- Usually the automatic tuning procedures result in high gains for optimal system response. This results also in axis vibration which should be removed by lowering the gains manually about 10% to 20%.
- The results for *following error* and *velocity error* should not be written to the parameters' file, to avoid irrelevant nuisance run-time motion controller errors.

7.2. Testing Procedures

Testing procedures for the system's first startup are contained in a separate document *MOSAIC - Test Procedure for System Startup*.

Additional testing procedures are to be defined for assessing performance parameters like accuracy, repeatability, stability, etc. These will typically entail special measuring equipment yet to be purchased. It should be noted, though, that the focus of the project is on developing a generic controller concept, rather than a commercial product, and therefore performance will sometimes be compromised for versatility.

8. PROGRAM PLANS

This section contains our short- and long-term aquisition and research plans.

8.1. Long Term

The program is planned to have four phases, from phase 0 to phase 3. Phase 0 (the first eight months) is concerned with the integration of the basic single-machine environment, which is described in detail in this document. The next three phases, each spanning a full year, are concerned, respectively, with multi-device machine, multi-machine, and the intelligent machine. Refer to the following table. More information may be found in our NSF proposal *The Design and Implemetation of an Open-Architecture Machine-Tool.*

Date and Phase	**Phase Description and Activities**
Nov.1 ’88 - Jun.30 ’89 Phase 0	**THREE-AXIS-SINGLE-MACHINE PHASE** • Integration of basic hardware for 3-axis machine • Initial work on Real-Time Operating System • Software integration to include the APT interpreter
Jul.1 ’89 - Jun.30 ’90 Phase 1	**MULTI-DEVICE-SINGLE-MACHINE PHASE WITH REAL-TIME OPERATING SYSTEM** • General evaluation of Phase 0 achievements • Further integration of operating system including the use of HIC and SAGE • Redesign motion controllers for interdependent motion • Demonstrate concurrent operation of the machine tool and Dexman • Develop version 1 of a manufacturing language modelled on APT and AML
Jul.1 ’90 - Jun.30 ’91 Phase 2	**MULTI-MACHINE PHASE WITH MANUFACTURING LANGUAGE** • Develop user-friendly production oriented front-end • Install a MAP-TOP Local Area Network • Develop version 2 of a manufacturing language that incorporates APT, AML and a machine-machine language such as CML
Jul.1 ’91 - Jun.30 ’92 Phase 3	**INTELLIGENT-MACHINE PHASE** • Use of open-architecture in case-study projects, e.g. • Use of active vision • Real-time communication with other sensors • Incorporation of expert systems on setup with CAD databases and the manufacturing language • Real-Time replanning of a setup.

Throughout the program efforts will be directed towards collaboration with other institutions involved in the development of software and hardware for advanced controllers. In particular, the Next Generation Controller (NGC) and the Intelligent Machining Workstation (IMW) should be considered.

The NGC program entails hardware and software design over the span of 6 years and $100 million. It covers a wide scope of controller functions and architecture, and proposes a development and implementation plan. Recently, an RFP (Request For Proposal) has been issued. Also, the National Center for Manufacturing Sciences (NCMS), a consortium of about 100 companies supporting research in manufacturing technologies, has been working with the U.S. Air Force on the NGC. The program emphasizes the open-system design, and suggests that the controller will be configured as a modular system, the components of which will have a standard interface, and will be manufactured by the various cooperating companies. It is not clear whether the program will stimulate the participation of the computer industry, or which computer standards will be adopted. The consortium solution seems not be the best way to efficiently exploit the capabilities embedded in the wide structure of the computer industry. Hopefully, concepts like MOSAIC, which rely heavily on the computer industry, will contribute to this important program.

The IMW program is concerned predominantly with intelligent software development, assuming the use of mainstream current controller hardware design, along with projected controller improvements such as a MAP interface to allow dual-way communications. This program is also supported by the U.S. Air Force, and is carried out as a collaboration between Cincinnati Milacron company, Pratt & Whitney company and Carnegie-Mellon University. The program includes the development of a family of expert systems for planning and cutting (planner, cutter, holder, sensing), languages for part description (MFGMAP) and feature exchange (FEL) in conjunction with a solid modeler, and a supervising control and human interface. Some uncertainty as to the hardware configuration exists, which slows down the design and implementation of the control scheme. This is were the MOSAIC hardware may come into play. Most of the software developed under this program is expected to be portable, and may be used in projects like MOSAIC and others.

8.2. Short Term

This section provides our plans for the first half of the second phase (Phase 1). It includes full details of our prospective acquisitions and implemetation work and some of the considerations involved in our decisions. It covers the six month time frame from May - October 1989. A PERT chart for this period is included in figure 7.1. At the end of this time period the following goals should be achieved:

(1) The interpreter will understand a larger set of APT and sudo APT commands. This includes the ability to perform interpolations, transformations, and closing the force loop. In addition basic control of dexman and the vision processing card should be demonstrated.

(2) The operating system will be able to handle two way communication between the VMEbus and the Sun, The system will support interrupts, multi-tasking and multiprocessing.

(3) Dexman will be integrated into the system and controlled through the Sun.

(4) A vision system will be integrated in MOSAIC's environment.

The work is broken down in to seven sections: orders and integration, software enhancements, operating system, dexman, vision, papers and documentation, and student projects.

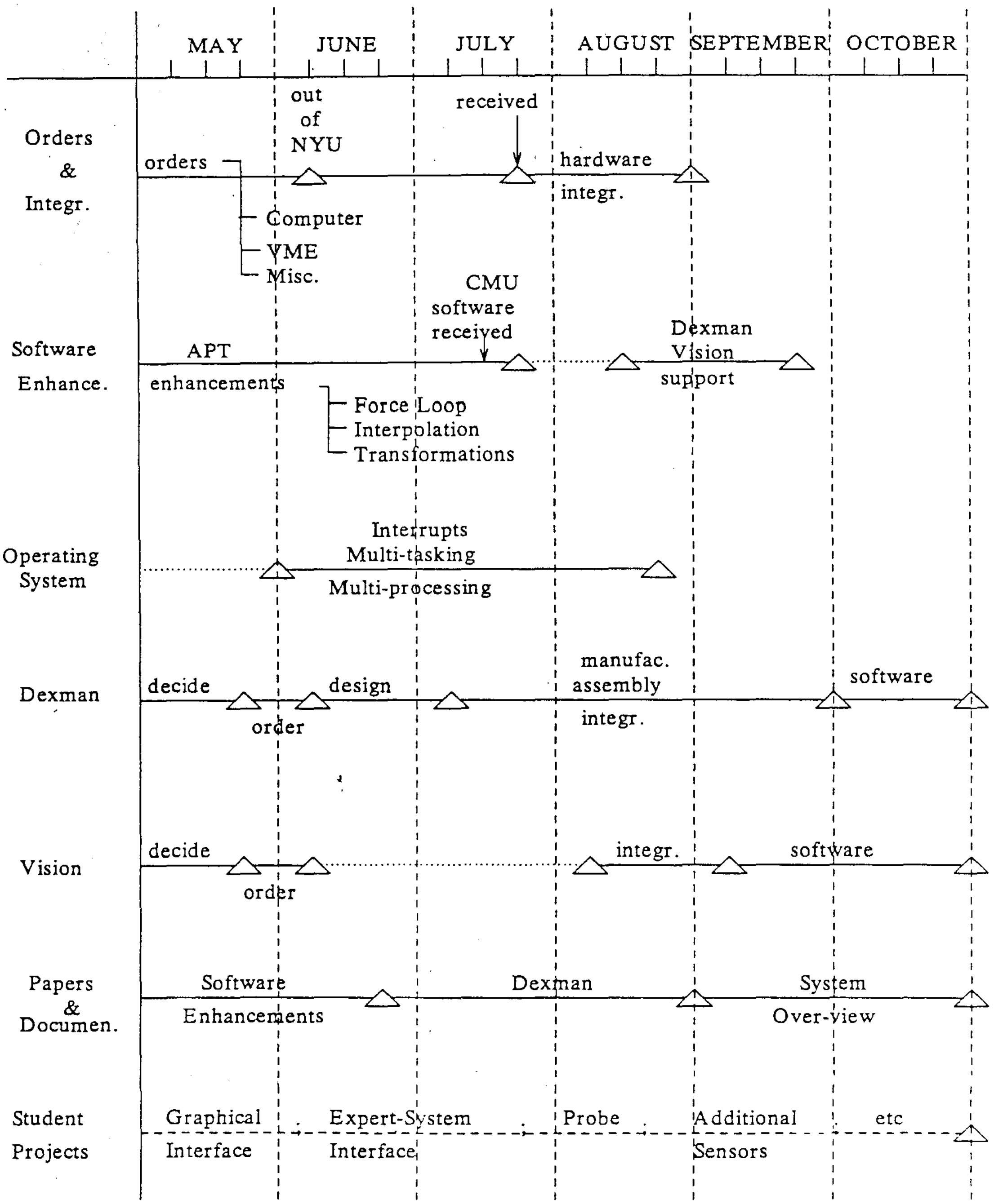


Figure 8.1. Short-Term Plan

Orders and Integration

The purchasing of components for integration into MOSAIC fall into five categories. The first three, computer, VME and miscellaneous are covered here. The components that are to be purchased for vision and dexman are covered in their respective sections. The computer components consist of disk and a SCSI controller ($5500 from Americad), and Sun memory (4MB @ $2800 from Americad). The disk and memory expansion are being purchased for two reasons. First the disk drive will provide room for storing images from the vision system as well as providing room for any software obtained from the IMW initiative. The memory will provide the power needed to drive this software as well as software expansions developed here. For the VME-Bus a CPU, VME memory (4MB), 3 motion controllers and I/O board(s) will be purchased. The CPU and memory will be used for multi-processing. We are currently considering The purchase of Pacific Micro 68020 (~$3800), Ironics IV-9001 ($NA) or the Mizar MZ 7170(~$7000). The CPUs from Ironics and Mizar are RISC based processors providing speed as well as special I/O facilities suited for multitasking. Three motion controllers will be purchased from Creonics ($6500) for use in controlling dexman. I/O board(s) will also be purchased to provide an interface for both digital and analog sensors, valves, etc. The miscellaneous components consist of cables and connectors needed to integrate the components.

Software Enhancements

The software enhancements fall into two categories: Standard APT and SUDO-APT commands. We would like to support a larger subset of standard APT commands such as circle and transformations. These commands will enable the machine tool to do more than the point to point linear moves that are currently supported. In addition we would like to support a few SUDO-APT commands such as closing the force loop and performing surface fitting to demonstrate how MOSAIC capabilities may be easily enhanced.

Operating System

The operating system will be enhanced with the following capabilities: Interrupt handling, Multi-processing, Multi-tasking and two way file communication.

Dexman

Dexman is essential to the development of a self sustaining machine tool. The ability to load and unload stock and tools as well as to maintain a clean working environment are the minimum requirements for a self sustaining machine tool. This ability is to be achieved in either of two ways, the purchase of a small 6 degree of freedom robot dedicated to the machine tool or the refurbishment of the RS2 wrist for use in this application.

Vision

Vision is an essential component of the MOSAIC environment when the project is looked at as a whole. What is provided in terms of capabilities and function are outweighed by the cost of a complete vision system (real time vision board and software) at this time. However if a vision system is to be purchased for general use in the lab and its cost is distributed in this way a vision system is desirable. It will provide MOSAIC with an essential sensing device, reduce its cost in terms of the project and will provide an incentive for members of the lab with a strong background in vision to aid in the project. The following vision systems are being considered:

HARDWARE:

Matrox VIP 1024 frame grabber ($2745)
Matrox MVP-VME frame grabber/processor ($6065)
Imaging 150 series ADI,FB,sync ($6000)
Imaging 150 series ADI,FB,ALU ($8000)

SOFTWARE:

KBVision from AAI ($7800-11800)
Visilog from Noesis ($8255 +$1500 for shape option)

The minimum requirements for a vision system include a frame grabber and basic software package. This will allow for non-real time vision. Two frame grabbers are indicated above the VIP 1024 and the 150-ADI, FB and sync. The VIP 1024 is non-expandable but the series 150 products are. In terms of software the KBVision does not directly support any VME-boards but it can be mabe to work with the boards. Visilog does support these VME-based vision boards.

An alternative to purchasing the software specified above is to purchase turn-key vision applications. Mnemonics (Michael Negin) can provide such systems. His company can also provide development software TIPS which was developed for PC market but has been ported to the sun for Rutgers.

Papers and Documentation

The work completed should be documented and papers describing what has been accomplished should be written. A paper describing the software enhancements i.e. graphical interface, APT support, multi-processing and multi-tasking capabilities should written. This paper will outline the interaction with a machine tool and stress its ability to be easily enhanced and modified as a result of the environment. A paper on Dexman, its development, integration and implementation should also be written. This paper will stress Dexman's functionality in the development of a self sustaining machine tool. Finally a paper providing a system over-view should be written. This paper will touch upon what has been accomplished as well as what is in the future (i.e. the development of a manufacturing language/environment).

Students Projects

This section includes the development of the Graphical interface, Expert-System, probe, additional sensors, etc. The completion of these tasks depends on finding students to work on these projects. If they are not completed the task will be moved into the next time frame.

APPENDIX A: COMPONENTS WIRING PIN CONNECTIONS

This appendix contains detailed wiring lists of the main components and their placement.

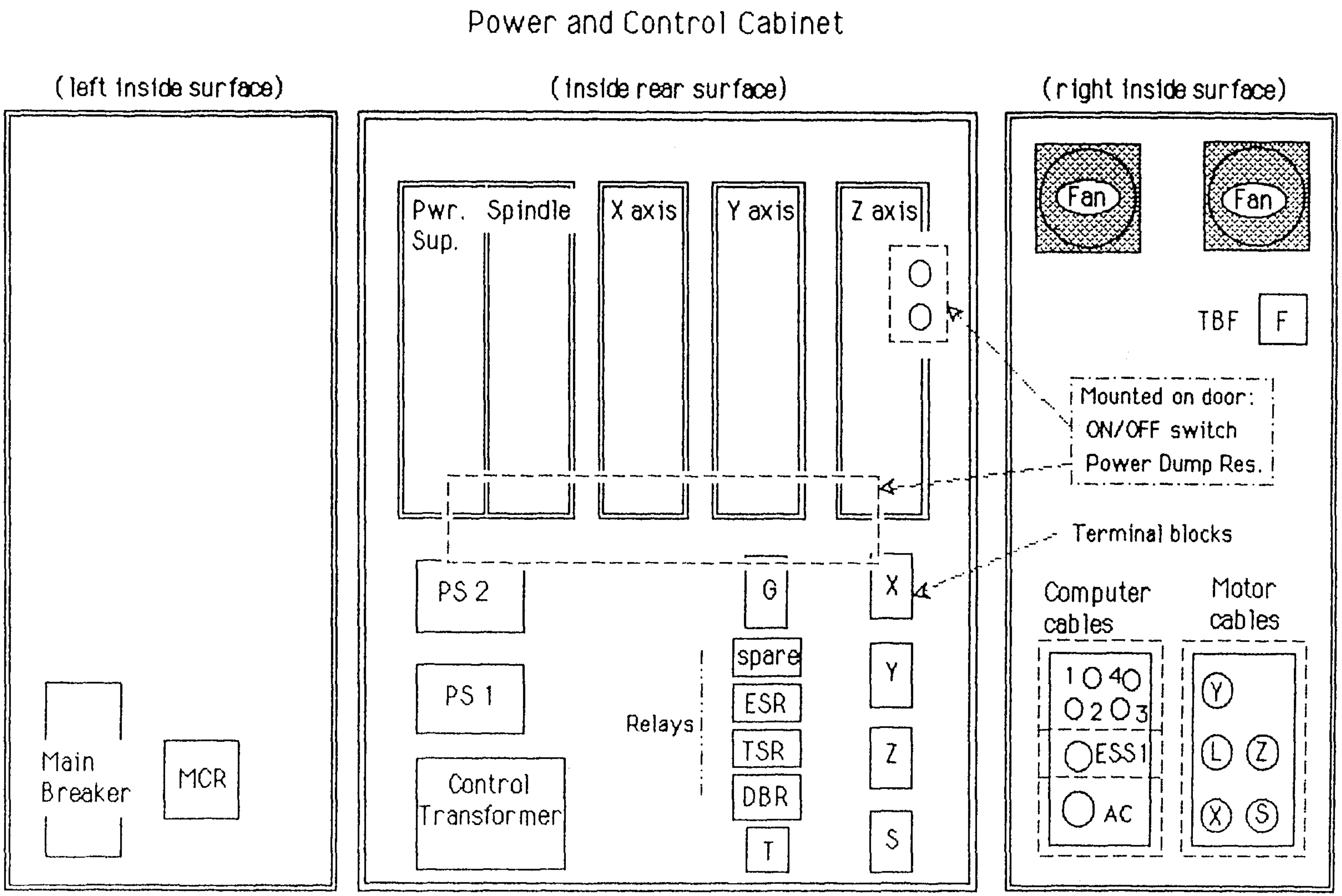


Figure A.1. Electrical Cabinet Component Placement

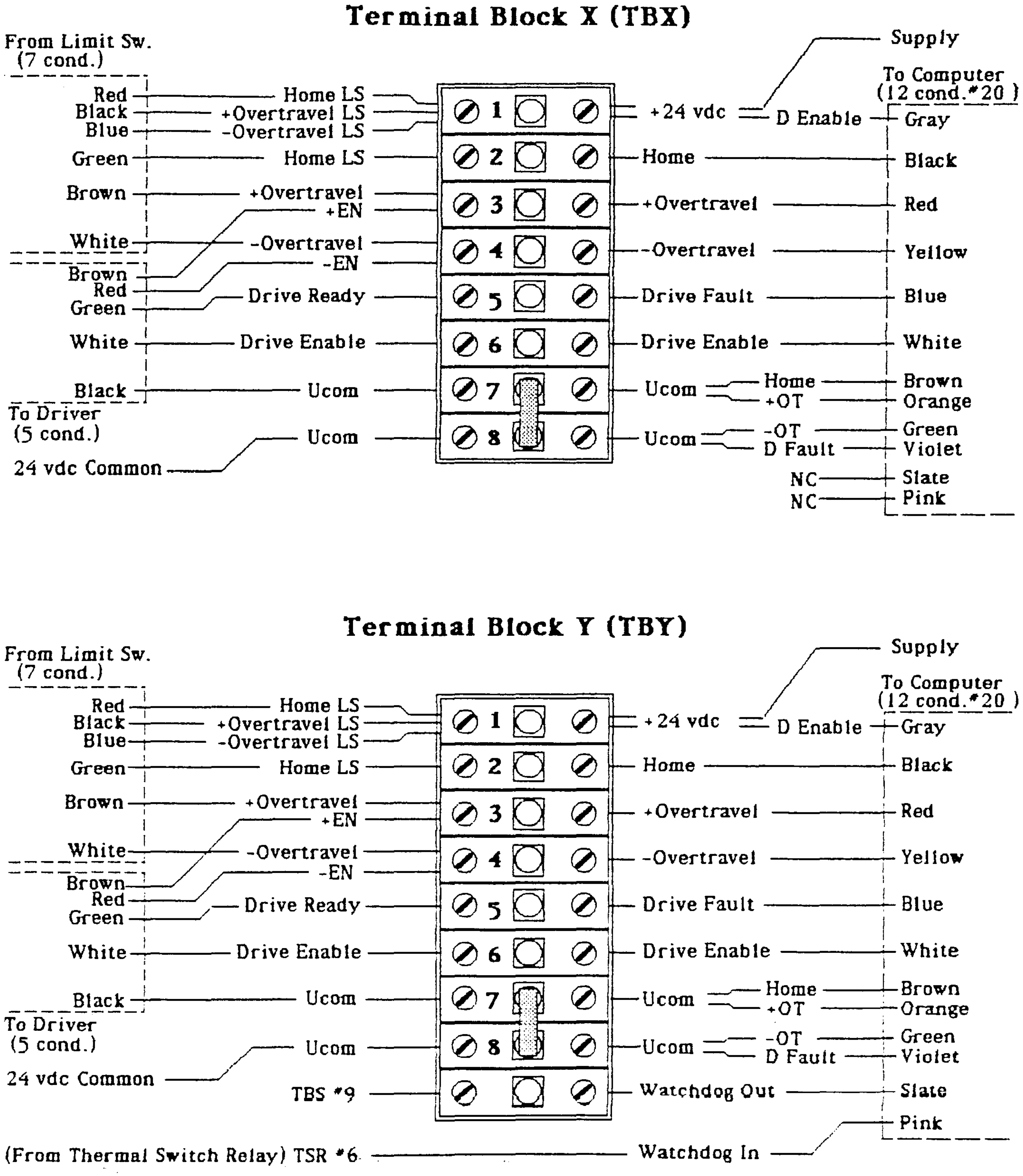
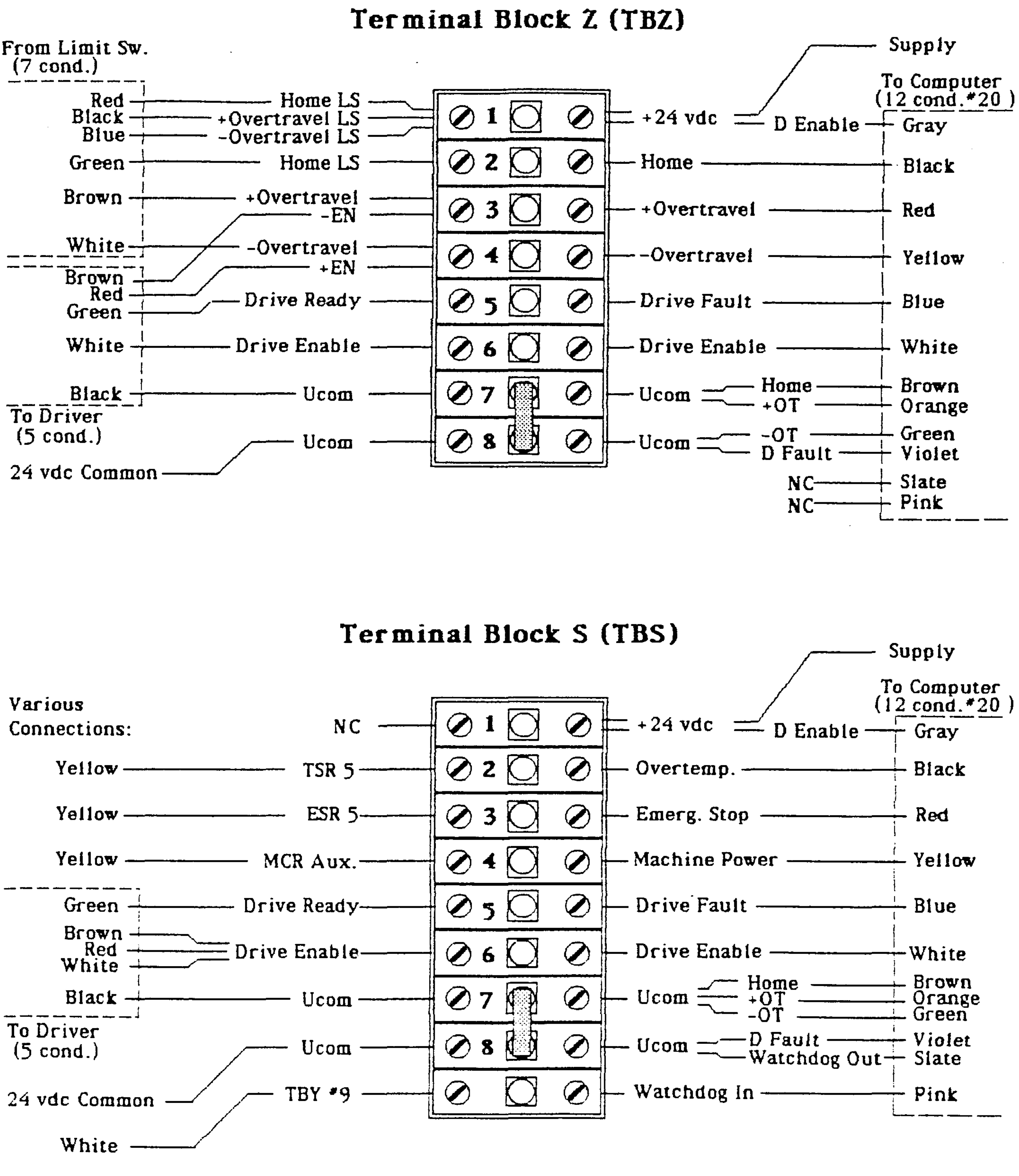


Figure A.2. X and Y Terminal Blocks Connections



Terminal Block Z (TBZ)

Terminal Block S (TBS)

Figure A.3. Z and S Terminal Block Connections

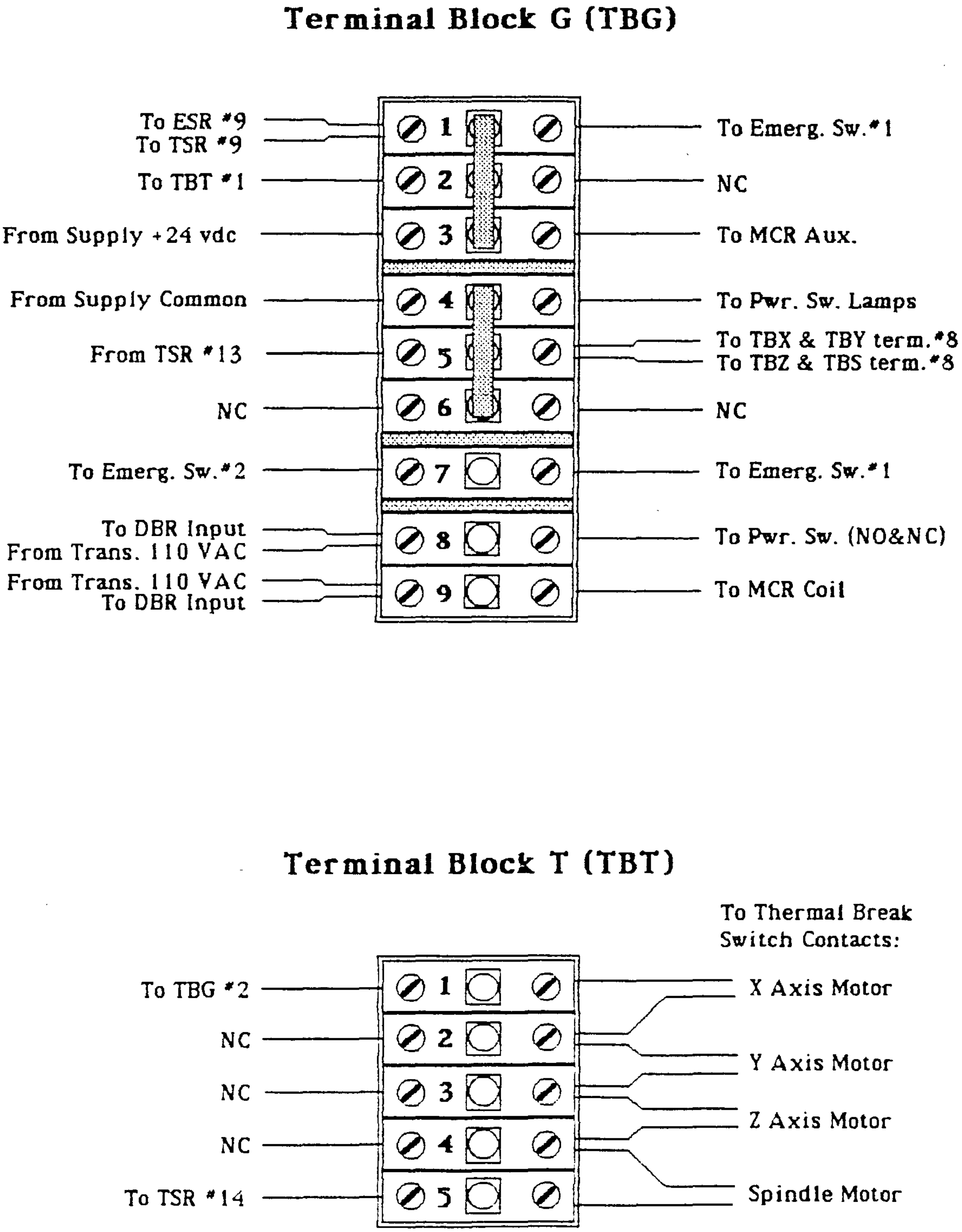


Figure A.4. G and T Terminal Block Connections

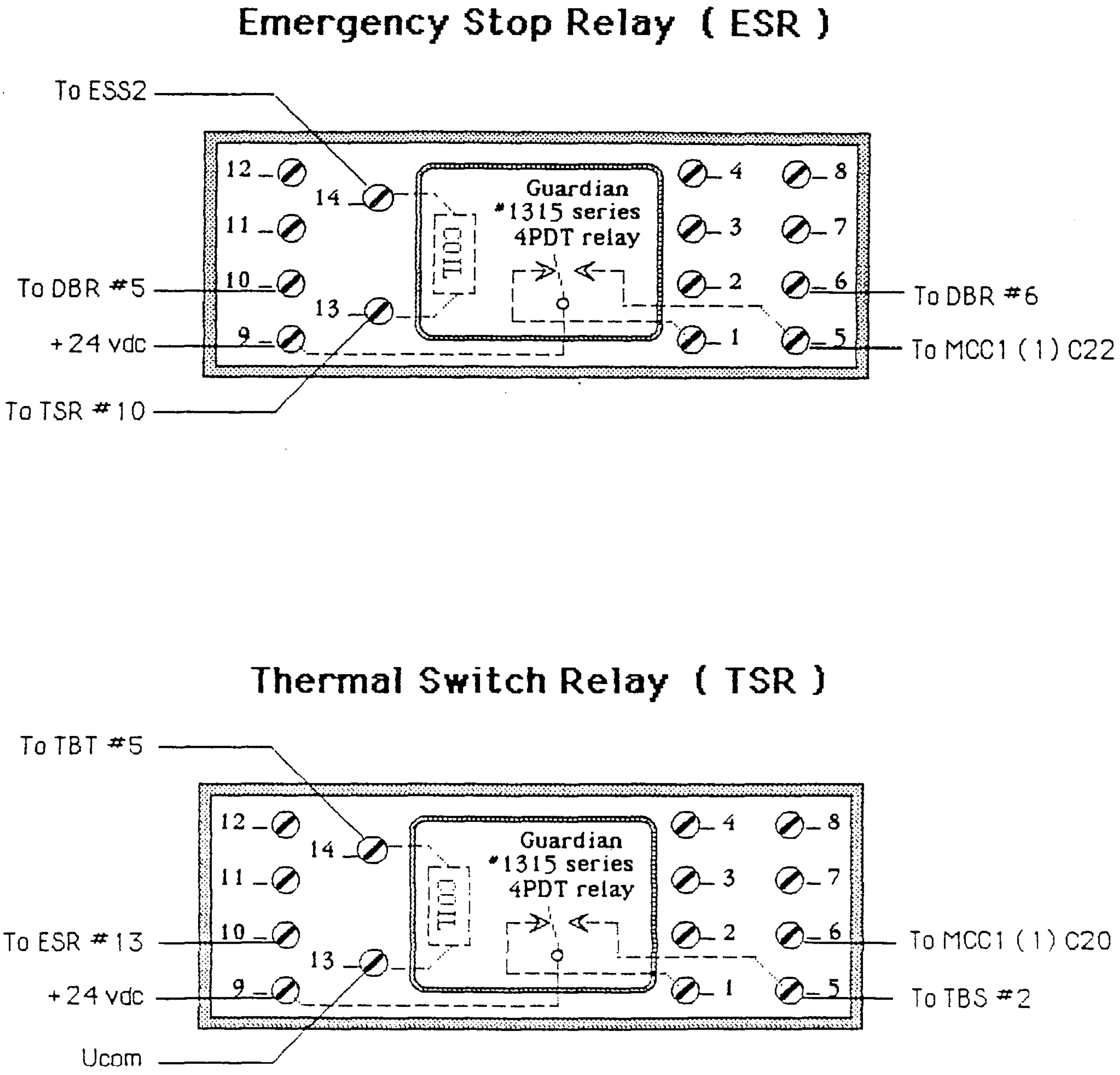


Figure A.5. Emergency and Thermo Relay Terminal Connnections

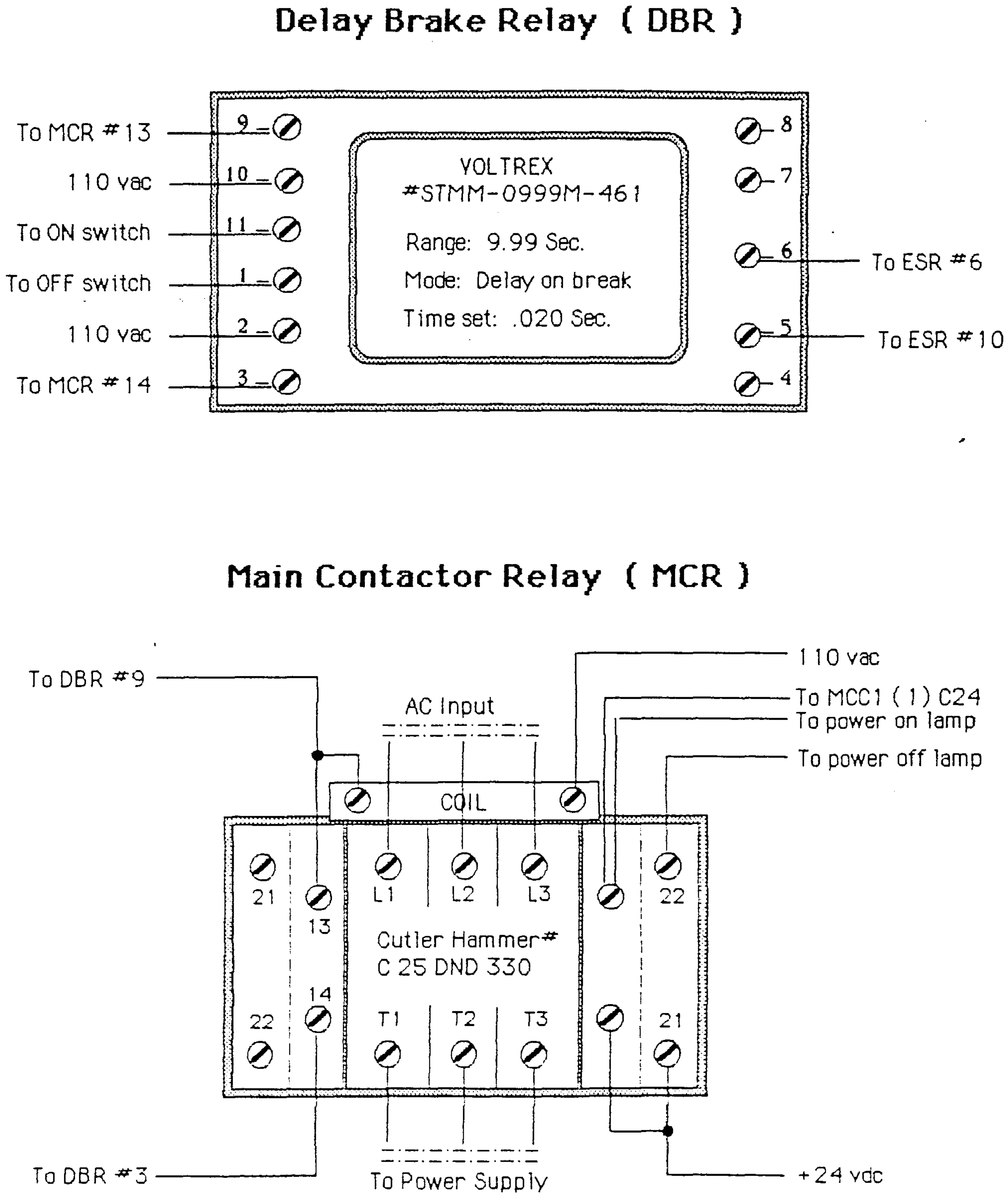
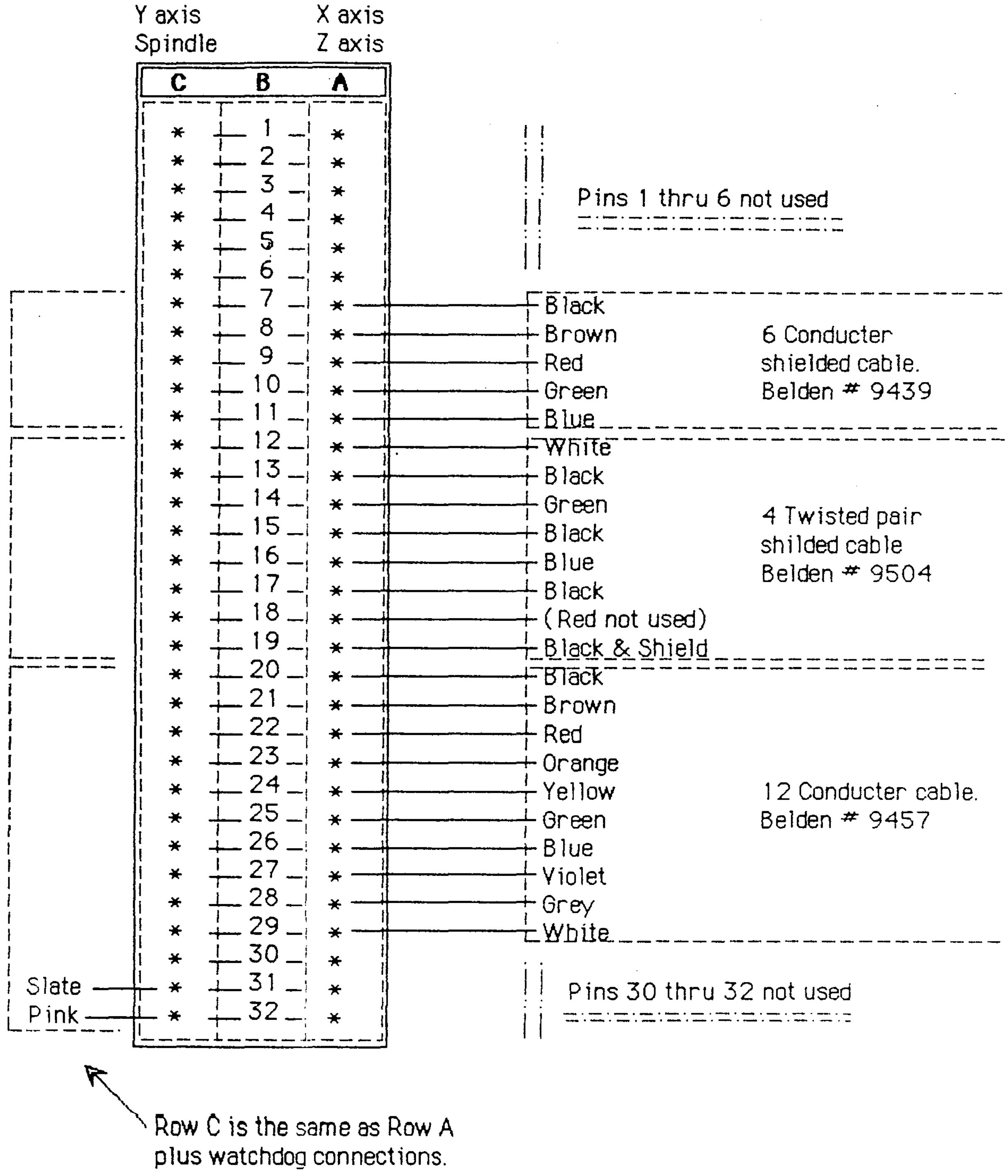


Figure A.6. Contactor and Time-Delay Relay Terminal Connnections



Row C is the same as Row A
plus watchdog connections.

View is from rear of VME Rack.

Figure A.7. Creonics Motion Controllers Pin Connections

APPENDIX B: CREONICS MOTION CONTROLLERS PARAMETERS SETTINGS

X AXIS

```
# Gains for x-axis
#            Proportional Integral  Velocity
mccommand x 7 0  0x85 0x2a  0x6e 0  0x18 0

# Home position for x-axis
mccommand x 7 1    0 0 0 0

# Home and overtravel mode for x-axis
mccommand x 7 2    0x3b 3

# Maximum positive travel limit for x-axis
mccommand x 7 3    0xff 0xff 0xff 0x7f

# Maximum negative travel limit for x-axis
mccommand x 7 4    0 0 0 0x80

# Maximum velocity and acceleration
#                     Vel
mccommand x 7 5  0x10 0x5c 0x15 0
  Acc         Dec
0x72 0x1c 1 0  0x32 0xb3 1 0

# Following error tolerance for x-axis
mccommand x 7 6    0xff 0x7f

# Deadband compensation for x-axis
mccommand x 7 7    3 0

# Feedfoward gain for x-axis
mccommand x 7 8    0x45 0x22

# Encoder mode for x-axis
mccommand x 7 9 0x1c

# Axis and drive type for x-axis
mccommand x 7 0xa 0

# In-position tolerance for x-axis
mccommand x 7 0xb 7 0

# Velocity error tolerance for x-axis
mccommand x 7 0xc 0x96 0x51 0xe 0xf

# Unwind constant for x-axis
mccommand x 7 0xd 0xa0 0xf 0 0
```

Y AXIS

```
# Gains for y-axis
#            Proportional  Integral   Velocity
mccommand y  7 0  0xff 0x2f  0x8e 4  0x4c 0

# Home position for y-axis
mccommand y  7 1    0 0 0 0

# Home and overtravel mode for y-axis
mccommand y  7 2    0x3b 3

# Maximum positive travel limit for y-axis
mccommand y  7 3    0xff 0xff 0xff 0x7f

# Maximum negative travel limit for y-axis
mccommand y  7 4    0 0 0 0x80

# Maximum velocity and acceleration
mccommand y 7 5  0x10 0x5c 0x15 0
0x72 0x1c 1 0  0x32 0xb3 1 0

# Following error tolerance for y-axis
mccommand y  7 6    0xff 0x7f

# Deadband compensation for y-axis
mccommand y  7 7    3 0

# Feedfoward gain for y-axis
mccommand y  7 8    0 0x24

# Encoder mode for y-axis
mccommand y  7 9 0x1c

# Axis and drive type for y-axis
mccommand y  7 0xa 0

# In-position tolerance for y-axis
mccommand y  7 0xb 7 0

# Velocity error tolerance for y-axis
mccommand y  7 0xc 0x96 0x51 0xe 0xf

# Unwind constant for y-axis
mccommand y  7 0xd 0xa0 0xf 0 0
```

Z AXIS

```
# Gains for z-axis
#               Proportional  Integral   Velocity
mccommand z  7 0  0xff 0x3b   0x8d 2   0x32 0

# Home position for z-axis
mccommand z  7 1      0 0 0 0

# Home and overtravel mode for z-axis
mccommand z  7 2      0x3b 3

# Maximum positive travel limit for z-axis
mccommand z  7 3      0xff 0xff 0xff 0x7f

# Maximum negative travel limit for z-axis
mccommand z  7 4      0 0 0 0x80

# Maximum velocity and acceleration
mccommand z 7 5   0x10 0x5c 0x15 0
0x72 0x1c 1 0   0x32 0xb3 1 0

# Following error tolerance for z-axis
mccommand z  7 6      0xff 0x7f

# Deadband compensation for z-axis
mccommand z  7 7      5 0

# Feedfoward gain for z-axis
mccommand z  7 8      0 0x24

# Encoder mode for z-axis
mccommand z  7 9      4

# Axis and drive type for z-axis
mccommand z  7 0xa    0

# In-position tolerance for z-axis
mccommand z  7 0xb 7 0

# Velocity error tolerance for z-axis
mccommand z  7 0xc 0x96 0x51 0xe 0xf

# Unwind constant for z-axis
mccommand z  7 0xd 0xa0 0xf 0 0
```

SPINDLE

```
# Gains for spindle
#            Proportional Integral  Velocity
mccommand s 7 0   0x42 0x33   0x0 0   0x35 0

# Home position for spindle
mccommand s 7 1     0xd3 0xfe 0xff 0xff

# Home and overtravel mode for spindle
mccommand s 7 2      0x13 3

# Maximum positive travel limit for spindle
mccommand s 7 3      0xff 0xff 0xff 0x7f

# Maximum negative travel limit for spindle
mccommand s 7 4      0 0 0 0x80

# Maximum velocity and acceleration
#                        Vel
mccommand s 7 5   0x10 0x5c 0x15 0
   Acc          Dec
5 0x75 1 0   0x32 0xb3 1 0

# Following error tolerance for spindle
mccommand s 7 6      0xff 0x7f

# Deadband compensation for spindle
mccommand s 7 7      0 0

# Feedfoward gain for spindle
mccommand s 7 8      0 1

# Encoder mode for spindle
mccommand s 7 9    4

# Axis and drive type for spindle
mccommand s 7 0xa 0

# In-position tolerance for spindle
mccommand s 7 0xb 7 0

# Velocity error tolerance for spindle
mccommand s 7 0xc 0x96 0x51 0xe 0xf

# Unwind constant for spindle
mccommand s 7 0xd 0xa0 0xf 0 0
```

REFERENCES

1. I. Greenfeld, F.B. Hansen and P.K. Wright, *Self-Sustaining Open-System Machine Tools,* NYU - Courant Institute of Mathematical Sciences, NYC, NY (September 1988). to be presented at NAMRC XVII

2. P.K. Wright and I. Greenfeld, *Open-Architecture Manufacturing - The impact of Open-System Computers on Self-Sustaining Machinery and the Machine Tool Industry,* NYU - Courant Institute of Mathematical Sciences, NYC, NY (May 1989).

3. I. Greenfeld, *Open-System Machine Controllers - The MOSAIC Concept and Implementation,* NYU - Courant Institute of Mathematical Sciences, NYC, NY (May 1989).

4. I. Greenfeld and P.K. Wright, *A Generic User-Level Specification for Open-System Machine Controllers,* NYU - Courant Institute of Mathematical Sciences, NYC, NY (May 1989).

5. P.K. Wright and I. Greenfeld, *The Design and Implementation of an Open-Architecture Machine-Tool,* NYU - Courant Institute of Mathematical Sciences, NYC, NY (February 1989). A proposal submitted to NSF

6. U.S. Airforce, *Next Generation Machine-Workstation Controller,* Materials Laboratory, Air Force Wright Aeronautical Laboratories, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio, 45433-6533 (November 1987).

7. Cincinnati Milacron Marketing Company, *Intelligent Machining Workstation Initiative (IMW),* Materials Laboratory, Air Force Wright Aeronautical Laboratories, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio, 45433-6533 (1988). Technical Reports prepared under Air Force Contract #F33615-86-C-5038

8. J.U. Korein et al., "A configurable System for Automation Programming and Control," *Proceedings of IEEE Conference on Robotics and Automation*, IEEE, (April 1986).

9. P.K. Wright and D.A. Bourne, *Manufacturing Intelligence,* Addison-Wesely, Reading, MA (1988).

10. I.H. Kral, *Numerical Control Programming in APT,* Prentice-Hall, Englewood Cliffs, NJ (1986).

11. B.L. Jones, *Introduction to computer Numerical Control,* John Wiley & Sons, New York, NY (1986).

12. D.G. Johnson, *Programmable Controllers for Factory Automation,* Marcel Dekker, Inc., NYC, NY (1987).

13. J. Tal, *Motion Control by Microprocessors,* Galil Motion Control, Palo Alto, CA (April 1988).

14. J. Tal, *Motion Control Applications,* Galil Motion Control, Palo Alto, CA (1989).

15. L. Salkind, *The Sage Operating System,* NYU - Courant Institute of Mathematical Sciences, NYC, NY (September 1988). Technical report No. 401

16. L.R. Nackman, et. al., *AML/X: A Programming Language for Design and Manufacturing,* IBM Thomas J. Watson Research Center, Yorktown Heights, NY (July 1986). Research Report 11992 (#54006)

17. D.A. Bourne, "CML: A Metta Interpretor for Manufacturing," *AI Magazine* 7 No.4 pp. 86-96 (1986).

18. F.B. Hazen and P.K. Wright, "An Autonomous, Self-Loading Machine Tool," *Proceedings of the Manufacturing '88 Conference*, American Society of Mechanical Engineers, (1988).

19. C. Hayes and P.K. Wright, "Automated Planning in the Machining Domain," *Knowledge Based Expert Systems for Manufacturing* **24** pp. 221-232 American Society of Mechanical Engineers, (1986).

20. A. Goldberg and D. Robson, *Smalltalk-80: The Language and Its Implementation*, Addison-Wesley, Reading, MA (1983).

21. U.S. Department of Defense, *Ada Reference Manual*. July 1980.

22. B. Stroustrup, *The C++ Programming Language*, Addison-Wesley, Reading, MA (1986).

23. Hartford, *Turret Millers - Installation, Operation and Maintenance Manual*, Hartford, Teterboro, NJ (1988).

24. General Motion, *TrueServo Motors and Amplifiers Instruction Manual*, General Motion Company, Minnetonka, MN (November 1988). Document No. 117150, Rev. 9

25. General Motion, *SD Spindle Drives Instruction Manual*, General Motion Company, Minnetonka, MN (November 1988). Document No. 117119, Rev. 9

26. Sun Inc., *Sun 3/160 Hardware Installation Manual*, Sun Microsystems, Inc., Mountain View, CA (1987-6).

27. HVE, *VMEbus Repeater 2000 - Product Specification AS90172000A*, HVE Engineering, Inc., Campbell, CA (June 1987). Part No. UM90172000A

28. Pacific Microcomputers, *PV682 User's Manual*, Pacific Microcomputers, Inc., San Diego, CA (July 1987). Document No. 022-1778-001, Rev. B

29. Creonics, *VMEbus Compatible Motion Control Card User's Manual*, Creonics, Inc., Lebanon, NH (March 1987). Stock No. 999-041, Rev. D

30. I. Greenfeld, *MOSAIC - Test Procedure for System Startup*, Courante Institute of Mathematical Sciences , NY, NYC (February 1989).

31. Sun Inc., *Sun Unix 3.5*, Sun Microsystems, Inc, Mountain View, CA (1987). 11 Vols.

32. L. Salkind, *The Sage Operating System*, NYU - Courant Institute of Mathematical Sciences, NYC, NY (September 1988). Ph.D. Draft

33. Dayton Clark , *Hierarchical Control System* , Courante Institute of Mathematical Sciences , NY, NYC (February 1989). Technical Report No. 396

34. C. Weingartner, *Machinists' Ready Reference*, Prakken Publications, Ann Arbor, MI (1986).

35. Control Engineering, Microcomputer Interface Group, *VMEbus Buyers's Guide*, Cahners Publishing, Des Paines, IL (December 1988).

36. CarrLane Mfg. Co., *Tooling Component Library in 3-D Iges Format*, CarrLane Mfg. Co., St. Louis, MO (1988).

37. MCS Inc., *Anvil-5000 Grapl Library*, Manufacturing and Consulting Services, Inc., Irvine, CA (1988).

38. MCS Inc., *Anvil-5000, Release 1, Revision 1.8*, Manufacturing and Consulting Services, Inc., Irvine, CA (1987). Training, Installation and Operation Manuals (8 Vols.)

39. Synermation Co., *MFNC01 Post Processor for Kitamura MyCenter-1/485 Fanuc 10M Control*, Synermation Co. (1988).

40. Sun Inc., *Sun Common Lisp,* Sun Microsystems, Inc, Mountain View, CA (1986). 3 Vols.

41. C.L. Forgy, *OPS5 User's Manual,* Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PN (July 1981).

42. National Bureau of Standards, *Initial Graphics Exchange Specification (IGES),* U.S. Department of Commerce, National Technical Information Service, Washington, DC (April 1986).

43. World Federation of MAP Users Group, *MAP Reference Specification - Manufacturing Automation Protocol (based on GM MAP 2.2 Specification),* Society of Manufacturing Engineers, Dearborn, MI (June 1987).

44. Micrology pbt, *VMEbus Specification Manual,* Micrology pbt, Inc., Tempe, AZ (October 1985). Revision C.1

45. I. Greenfeld and Louie Pavlakos, *NYU Machining Cell Users' Manual,* NYU - Courant Institute of Mathematical Sciences, NYC, NY (June 1989). (Technical Report)

46. Kitamura Ltd., *Operation Manual Drilling Center M-1/485,* Kitamura Machinery Co., Ltd., Toide Takakoa, Japan ().

47. Fanuc Ltd., *Fanuc 10M-A Operator's Manual,* Fanuc Ltd. (1984-6). B-54824E/03 & B-54810E/01 (2 Vols.)

48. Renishaw Ltd., *User's Guide MP7-8-9 Probes With Compact Optical Transmission System,* Renishaw Metrology Ltd., England (1986). H/2000/5080/B

49. I. Greenfeld, *Probe Programming Users' Guide - NYU Machining Cell,* NYU - Courant Institute of Mathematical Sciences, NYC, NY (July 1988).

50. IBM, *IBM Robot System/1 - AML Concepts and User's Guide,* IBM (). Document No. SC34-0411-0, and associated documents GA34-0180 and SC34-0410