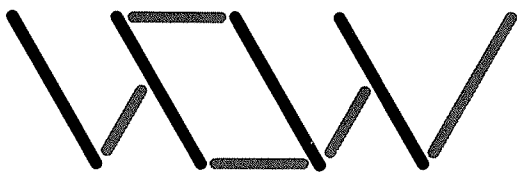
NSW

OWNER OPERATOR MANUAL



MG-1 Owner Operator Guide

Whitechapel Computer Works Ltd.,
75 Whitechapel Road,
London E1 1DU.

Table of Contents

Chapter 1
Preface

This MG-1 Owner/operator Guide is designed to take you through the setting up and use of your system. It is intended for use with other WCW documents, and with existing GENIX™ documentation. A pointer to these documents is given in Appendix F. This Guide should act as a basis for further reading.

The major features of the MG-1 are described. Particular attention is paid to its special features such as its interactive graphics capabilities and its Window Manager. Experienced users of operating systems derived from UNIX™ will be able to scan the Guide and by way of the Introduction, System Description and Index. Chapters 6 and 7 cover the basic points of GENIX. The Programmer's Manual will highlight differences between GENIX and similar operating systems.

For the new user, the System Description and the operating software chapters should be the most important. Chapter 8, on the interactive graphics, covers the MG-1's image handling and design capabilities, including the Window Manager. The System Administration and Security chapters are of immediate interest when the MG-1 is to be used by a single user.

The Owner/operator Guide is divided into twelve main sections:

1. Preface

2. Introduction.

The Introduction is a brief survey of the MG-1. The hardware and software are introduced; more detail is given in Chapter 3 and the GENIX chapters. The physical specifications of the system are given in Appendix A. The Introduction should provide enough information for a start to be made with the system.

3. System Description.

This section provides a more detailed coverage of the MG-1's major hardware and software features and can be read in conjunction with the MG-1 Technical Manual. The GENIX operating system is introduced, and attention is paid to the essentials of the interactive graphics system. To complete this description of the system elements of the MG-1, a list of the MG-1's optional features is given.

4. Installation.

In order to set up the MG-1, certain site conditions should be met. These relate to such details as proximity of power points, inter-unit leads, and the types of surface needed for the mouse. The information needed to set up the MG-1 and to install its optional features is clearly laid out. See also the Technical Specifications in Appendix A.

5. Getting Started.

This section is a guide to starting up the MG-1, replying to the initial operating system prompts, selecting start-up options, and closing down the system when the session is over.

UNIX is a Trademark of AT&T Bell Laboratories GENIX is a Trademark of National Semiconductor Corporation

6. GENIX — An Overview.

This chapter is an introduction to the GENIX operating system, and should be read with the next chapter. The basic features of the GENIX shells, the file and directory structures, the graphics and window handling, and the utilities are introduced. Full details of operating system facilities are available from the GENIX Programmer's Manual.

7. A GENIX Session.

In this section, the most commonly used GENIX commands are introduced. File and directory creation and handling are covered, and command combinations are illustrated. Full details of the operating system commands, the utilities and the C programming language are available in the GENIX Programmer's Manual.

8. Interactive Graphics.

The MG-1 is designed as a high performance graphics workstation, and offers some very powerful facilities. The Window Manager divides the screen into a number of "Virtual Terminals", each with its own specific job. For rapid graphics handling, the rasterop system system is available. These and other graphics methods are described in this section. A interactive techniques catalogue is also provided, in section 8.3.

9. System Administration.

GENIX offers some very powerful facilities for managing the MG-1. While the single user may not need all of these facilities, all users should know how to monitor disk space and use the backup and archiving systems. The role of the superuser is covered. Full details of these systems are given in the GENIX Programmer's Manual.

10. Security.

Each user, or group of users, may be granted or denied access to files. This section introduces the idea of limitations on access permission as well as of passwords and user-ID codes. The security duties of the superuser are discussed.

11. Troubleshooting.

This section introduces the most commonly used hardware and software problem-handling methods. The MG-1 Technical Manual and the GENIX Programmer's Manual contain additional material.

12. Appendices.

A. Physical Specifications.
B. Serial Port Pin Allocations.
C. MG-1 ASCII Codes.
D. Scan Codes.
E. Power On Tests.
F. Reading List.

This guide is not intended to be a comprehensive GENIX manual, especially in the case of the utilities. These are described in full in the relevant operating system documents. For a comprehensive list of documentation to be consulted, for both hardware and software, see the Reading List in Appendix F.

Chapter 2
Introduction

During the early 1970's, the concept of the computerised office began to take shape. In order to handle the work load involved, the mini-computer system dedicated to the single user was developed. These systems were known as "workstations" but their high cost prohibited their wide-spread use. With the advent of the microcomputer, truly personal systems were possible, but in designing such systems around the need for low cost, many of the more impressive facilities were lost. These facilities typically covered large-scale memory, high-resolution graphics, and fast response time.

Increasingly, the distinctions between microcomputer, minicomputer and mainframe have become blurred and finally, in 1984, the Whitechapel Computer Works MG-1 system has succeeded in unifying the ideals of 'micro' cost and 'super-mini' power.

Such a work station has to be as flexible and easy to use as a microcomputer, but have the data-handling, graphics and communications facilities of a much larger system. The definitive combination of these features is the MG-1.

To satisfy demand for large scale storage at low cost, the MG-1 provides a basic system of main memory and back-up devices which may be radically up-graded to increase capacity as and when required. The MG-1 uses a 32-bit processor and the GENIX operating system. GENIX is the National Semiconductor implementation of AT&T's Bell Laboratories 4.1bsd UNIX. UNIX is rapidly becoming the industry operating system standard. A major advantage of this operating system is that applications are widely portable between machines.

To reduce the cost and physical size of the system, the MG-1 is constructed around a single-board, microcomputer-style architecture, using the latest VLSI technology (Very Large Scale Integration); for all its processing power, the MG-1 can compete with many PC systems for price. The implications of this development are significant: users need no longer share minicomputers. Traditional time sharing operations can be replaced by distributed computing at an easily justifiable cost.

In short, the MG-1 Work Station offers startling cost-efficiency, compact size, user-flexibility, and a number of special features. These include interactive "rasterop" graphics and full graphics library, Window Manager, expansion card facilities via an IBM PC™ compatible mother board, and the Ethernet local area networking system.

Many of the problems of the less elegant earlier systems have been overcome by the MG-1. For example, graphics handling was often slow: the MG-1's raster system uses rectangular arrays of pixels which can be rapidly handled by their own dedicated hardware. The combination of GENIX and IBM PC expansion cards solves the problems of limited device availability. This option, and the Ethernet networking system which can combine up to 200 MG-1 stations, offers a very high level of site flexibility. To improve data handling, traditional keyboard input may be supplemented by the mouse which is a standard MG-1 feature. As a further improvement to the working environment, the MG-1 is virtually silent.

The MG-1 and its operating system is a truly powerful combination.

The major features of the system are as follows:

Display.

The display is a landscape format 1024×800 pixel system rated at 80 dots per inch. This provides a high quality image which, because of the MG-1's high image refresh rate (57 Hz), remains flicker-free. For rapid graphics handling, the MG-1 supports "rasterops". The idea of the raster is explained in section 3.5.1, and in Chapter 8. The display is a bit-mapped system refreshed from the main memory, and uses a similar memory paging system to the Memory

IBM PC is a registered trademark of International Business Machines.

Management Unit. Accordingly, the graphics uses the memory in the most efficient way possible.

Memory.

The MG-1 has a standard 0.5 Mbyte main memory expandable in 0.5 Mbyte or 2 Mbyte units to a maximum of 4 Mbytes or 8 Mbytes of real memory, held within the standard enclosure. This provides up to 16 Mbytes of paged Virtual Memory per program. This gives excellent data processing and graphics handling facilities, controlled by a highly sophisticated Memory Management system. The concept of a memory "page" is described in section 3.2.1.3.

Processor.

The MG-1 runs on a 32-bit National Semiconductor NS32016 processor (Central Processing Unit) rated at 8 MHz clock rate. Floating point data handling is standard, using the NS32081 Floating Point Unit. The NS32082 Memory Management processor handles all memory transactions from the CPU to provide a highly efficient demand paged memory system. The CPU runs a 24-bit address bus allowing a Virtual Memory capacity of 16 Mbytes (2^{24} addresses).

Disk System.

Standard features are a 10 Mbyte hard disk and a 0.8 Mbyte floppy disk system supporting 5.25" double-sided, double-density, soft-sector, 96 tracks per inch disks. A variety of larger hard disk options are available. As an option, a Streamer Tape system is available which replaces the floppy disk drive. GENIX offers full flar format archiving and library facilities for the down-loading of data from hard disk to floppy disk or streamer tape.

Operating System.

The MG-1 supports GENIX, a full UNIX system with Berkeley 4.1bsd enhancements. The operating system provides virtual memory handling, multi-tasking priority control, input and output operations, and user interaction. GENIX provides a large number of utilities such as text editors, C and Assembler programming, compilers, comprehensive on-line documentation (the extent of which depends on the hard disk in use), and interactive calculator facilities. Other programming languages such as FORTRAN 77 and Pascal, are optional. GENIX will support a wide range of business software packages such as advanced word-processors and spread-sheets, and the IBM PC expansion cards will allow GENIX access to one of the world's largest computing markets.

Keyboard.

The IBM PC-compatible QWERTY-style keyboard is a standard feature of the MG-1 providing 83 keys, including 10 programmable function keys, and adjustable typing angle.

Mouse.

The mouse is a standard MG-1 feature and allows user manipulation of screen images and windows.

Expansion Ports.

Two expansion ports are provided. The memory expansion port will accept up to seven 0.5 Mbyte memory cards or up to three 2 Mbyte cards, thereby up-grading the standard memory to either 4 Mbytes or 8 Mbytes. The general purpose expansion port allows direct access to the buffered system bus, and allows installation of an IBM PC mother board which will hold up to three cards. The availability of IBM PC cards for the MG-1 opens up a huge range of facilities at low cost.

Communications.

The optional IEEE 802.3 Ethernet networking controller allows up to 200 stations to be linked together to create a communications and time-sharing network. Even large scale data objects may be rapidly transmitted as the system has a 10 Mbit per second transfer rate, and interfaces directly with the processor's address/data bus.

RS-232 C Port.

The single RS-232 port is a serial I/O interface between the MG-1 and such peripheral devices as line printers, plotters and other terminals. Software drivers allow the connection of further serial devices through the expansion card slots.

MG-1 GENIX.

The great majority of the operating system provided with the MG-1 is standard GENIX. Because the system is based on such revolutionary architecture, there are invariably some differences due to hardware factors. These are detailed in full in the GENIX Programmer's Manual. Experienced users should pay attention to the MG-1 Instruction Set, especially when using commands relating to the graphics facilities or the I/O devices. A number of other operations involve the use of commands specific to the MG-1, for example archiving. These commands are explained in full in Volume I of the GENIX Programmer's Manual.

The GENIX internal on-line documentation is available to any user with a hard disk configuration greater than 10 Mbytes. The full size version of the manual amounts to 1.4 Mbytes and is therefore only implemented on the larger systems. Chapter 11 of this manual lists the files containing the on-line manual and other 'expendable' material. These files may be deleted in order to free disk space.

Chapter 3
System Description.

3.1. Introduction.

The ideal workstation should incorporate a number of essential design principles: it should act as a general data processing system with a wide range of software for program design, text handling, and graphics; it should have a high degree of localised intelligence for the handling of specific tasks; it should be capable of implementing a creative design process involving more than one potential user; it should be responsive to the needs of the specific user without compromising larger scale objectives; it should be capable of handling data on many levels; and it should be physically robust enough to withstand user conditions that may be far from ideal.

The ways in which these principles have been built into the MG-1 will be covered in this section. An attempt has been made to describe many of the major concepts involved in a system of this complexity, but reference should be made to the Reading List in Appendix F for further details.

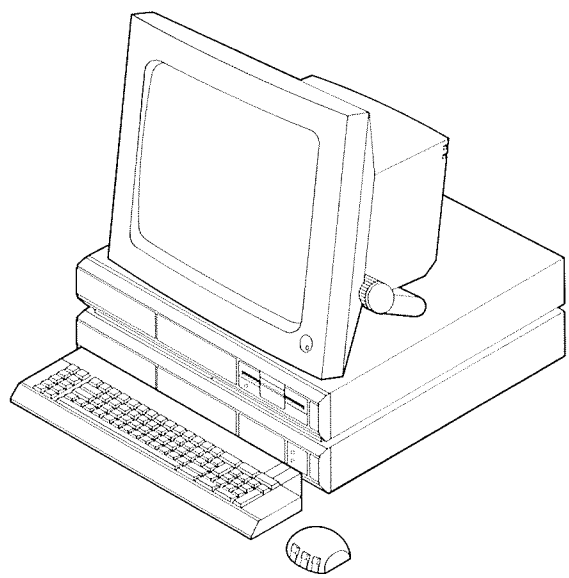


Figure 3.1. The MG-1 Personal Workstation.

3.2. The Hardware.

3.2.1. The Computer Unit.

The computer unit is the central component of the system, and contains all of the memory, processors, communications systems, expansion card facilities, Input/Output ports, and the power supply.

The unit as a whole measures approximately 50cms.×15cms.×45cms., and is strongly enough built to support the weight of the monitor unit. It may stand on its side where work surface space is limited.

Located on the rear panel of the unit are three expansion slots, and sockets for the IEEE 802.3 Ethernet system, serial RS-232 C port, the mains power supply, and the fuse holder. The single, quiet cooling fan is located inside this rear panel, and should not be obstructed.

On the right-hand side of the front panel is the Power-Up button. Above it is the Power-On LED (light-emitting diode). The power-down routine is handled under program control, so the machine cannot be switched off accidentally.

To the left of the Power-Up button is a flap covering the keyboard socket, the mouse socket, and the Diagnostic LED. The use and operation of this LED is described in Chapter 11.

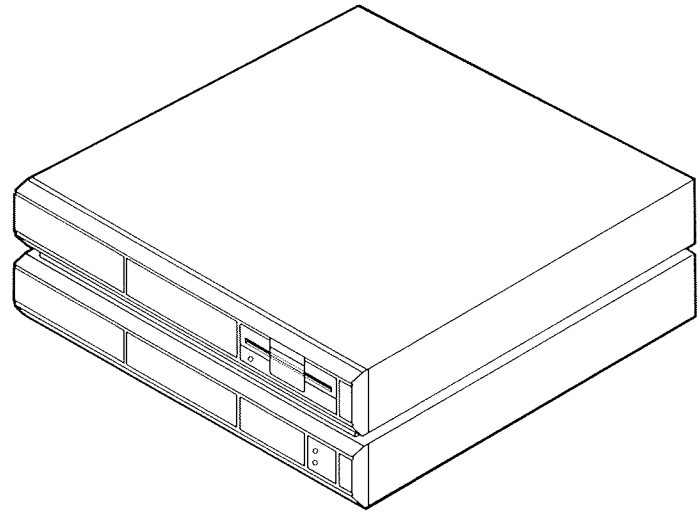


Figure 3.1. The Computer Unit.

Directly above this flap is the floppy disk drive or the tape streamer drive, depending on the configuration in use. Next to the floppy disk drive is the hard disk drive. The tape streamer and the hard disk systems have a range of capacities, both starting at 10 Mbytes.

The precise details are listed in Appendix A, Physical Specifications.

3.2.1.1. The Processors.

The MG-1's system logic is supported by the National Semiconductor Series 32000 chip set, comprising the following:

1. NS32016 Central Processing Unit (CPU). This processor is designed to support high performance multiprogramming, and controls all high speed priority switching and process control. It is designed to supply processing power to users needing large amounts of address space for large programs or data objects. The 32016 supports a 16 bit data and 24 bit address bus supplying 16 Mbytes of Virtual Memory per program.
2. NS32082 Memory Management Unit (MMU). This unit provides the hardware required to support a demand paged virtual memory system. The logical memory is organised as a series of pages which map either onto physical page frames or onto areas of memory held on peripheral memory devices such as disk drives. Peripheral memory pages are automatically swapped into main memory at a rate hardly detectable by the user.
3. NS32201 Timing and Control Unit (TCU). The TCU provides the system clock (rated at 10 MHz) and all processor control operations.
4. NS32081 Floating Point Unit (FPU). This extends the instruction set of the CPU to include fast 32 and 64 bit floating point operations.

In addition to the above, there are a number of processors dedicated to I/O functions, Direct Memory Access control, and Interrupt control. These are covered in more detail later.

A primary feature of the MG-1's processor set is the high degree of correspondence between the system architecture and high level language operation. There is a general lack of processor instructions and addressing modes that would interfere with the smooth running of high level compilers. In addition, the MMU includes hardware support for a major de-bugging tool: "breakpointing" halts a program at a particular instruction or data access point, and examines the state of the program to determine the cause of irregular behaviour.

3.2.1.2. Slave Processors.

The MG-1 controls many of its functions through two "slave processors" which allow operations that would not be permitted by conventional integrated architectures. These slave processors, which are unique to the Series 32000™ are the Memory Management Unit and the Floating Point

Series 32000 is a trademark of National Semiconductor Corporation.

Unit. Communication between the CPU and the slave processors occurs by way of a very fast protocol that remains transparent to the user.

The major advantages of this architecture relate to software adaptability. If these slave operations ever become integrated into the CPU through advances in hardware engineering, software developed under the slave system will still run, and in fact will run much more quickly. Software developed before the advent of the slave system can be run simply by removing the modules that were originally necessary to emulate the operations now performed by the slave processors.

Further details of the MG-1's processors can be obtained from references in the Reading List.

3.2.1.3. Memory Handling.

The Series 32000 chip set is based on a linear memory architecture, supports page-based mapping, and provides a demand-paged virtual memory system.

The total number of physically addressable memory locations (bytes) is the "physical address space". This is defined by the Main Memory physically available to the system; the MG-1 supplies up to 8 Mbytes of accessible Main Memory depending on the expansion cards in use. The number of locations that a program can possibly address is the "logical address space". In the case of a system such as the MG-1 which has a 24-bit address bus, this amounts to 16 Mbytes (2^{24} bytes). It is the logical address space that actually defines the memory architecture.

Where the individual bytes are accessed by consecutively numbered addresses, the memory is termed "linear". However, the bytes may be clustered into "segments"; when this is the case, each address consists of two parts, the segment address which defines the group of bytes involved, and the "displacement" or location of the actual byte within the segment. This type of memory organisation is well suited to the modern DP environment as modular programming relies on such segmented data handling systems.

The MG-1's NS32000-based memory is segmentally organised into 512 byte "pages". These are small enough to be flexible in the context of the variable sized segments used by programs, but large enough to offer an effective code protection system.

The action of translating a logical address into a physical address is called "mapping". Where the logical and physical address spaces are the same size, there is a direct relationship between a logical address and a physical address. In a multiprogramming environment, where processes operate concurrently, the user would have to 'manually' ensure that the memory dedicated to one process did not overwrite that dedicated to another.

Where the logical space is much larger than the physical space, as is the case with the MG-1, mapping operations are much more complex. The Series 32000 breaks the logical space of 16 Mbytes down into 32,768 pages, each of 512 bytes.

The physical address space is similarly broken down into a number of 512 byte units called "page frames". This means that a logical page can map straight a physical page frame. Each process initiated by the user has its own mapping table which contains the appropriate logical and physical addresses, and a code listing the protection attributes which limit access to areas of memory. This means that there is no danger of one program over-writing another.

The condition of the logical address space being larger than the physical address space is termed "Virtual Memory". This is an important feature of the MG-1, and gives the impression that the memory available to a program is much larger than is really the case. This is because logical memory pages are often mapped onto locations that are really held on peripheral memory devices such as disks. This process, called "page swapping", is necessary because while the 16 Mbytes logical memory contains 32,768 pages, a 4 Mbyte physical memory contains only 8,192 page frames. Thus only a fraction of the logical addresses may be mapped instantaneously and directly onto physical Main Memory locations at any one time. These mapping and page-swapping operations are handled by the 32082 Memory Management Unit, at a rate that is hardly detectable to the user.

The MG-1's NS32000 based memory is thus managed in such a way as to provide the highest degree of processing power, while reducing the cost to a minimum.

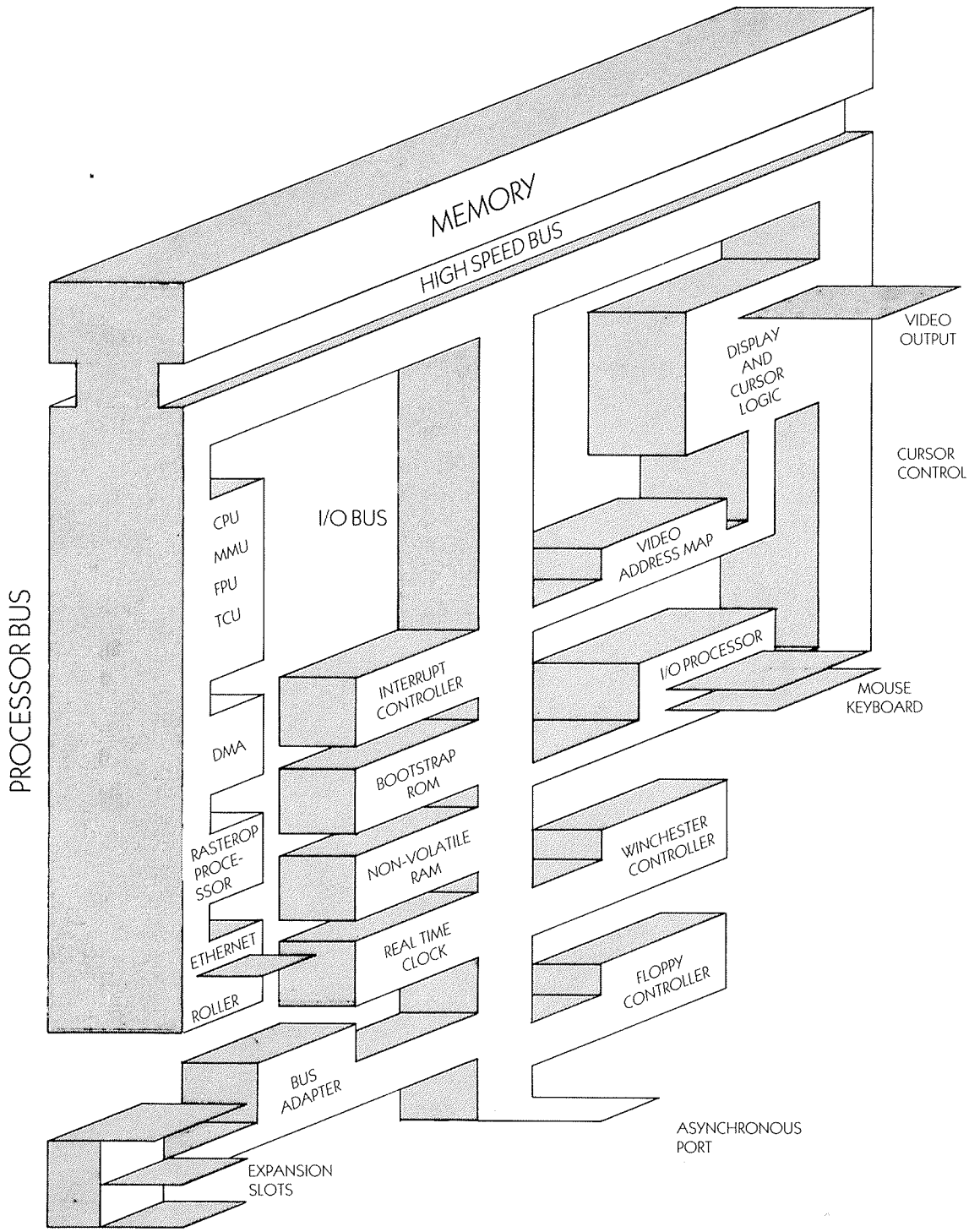


Figure 3.2. System Architecture.

3.2.1.4. Floating Point Operations Handling.

The NS32081 FPU is used to handle "real numbers" with fractional parts. Two sizes of floating point number can be represented by this processor, the 32 bit "single precision" type, and the 64 bit "double precision" type. Floating point numbers are held in the form:

$$\text{number} = \text{sign} \times \text{fraction (or mantissa)} \times 10^{\text{exponent}}$$

The advantage of using double precision floating point arithmetic is that the range of numbers is greater because of the larger exponent element, and the precision is greater because of the larger mantissa.

The FPU is 32 bits wide, and is broken down into a number of storage registers. Where double precision floating point representations are required, the number is held in two consecutive, catenated, registers.

3.2.1.5. Interrupt Handling.

The NS32202 Interrupt Control Unit (ICU) handles all conditions that alter the normal flow of processing events. Interrupts are events that occur externally to the central processor (and are distinct from "traps" which occur under program control), and are of two major types.

"Non-maskable" interrupts are initiated to preserve system integrity in the case of a catastrophe such as a power failure. They cannot be disabled, and have the highest priority. "Vectored" interrupts are assigned priorities and many are initiated by I/O devices, for example when a high speed device is talking to a low speed device. The slower device must deal with one set of data before accepting another, and so initiates an interrupt request.

A vector is a memory location containing the base address of an interrupt handling routine. The ICU handles interrupt requests according to the priorities assigned and the efficiency with which these operations are carried out largely determines the performance of the multiprogramming and I/O systems. This is especially the case as active programs are frozen according to the current state of the system. The CPU ensures that the system is unchanged when the program is re-activated.

3.2.1.6. Input/Output Handling.

The I/O system is the interface between the processors and the outside world, and in the MG-1 is highly integrated into the processor and memory system. The Series 32000 chip set supports a "memory mapped" I/O system in which peripheral input and output devices are considered to be a part of the memory so as to allow the full range of processor addressing modes.

The system works by assigning a number of registers to each I/O interface. These registers respond to read and write commands in exactly the same way as other areas of memory. Therefore tests or modifications on the contents of the I/O interfaces can occur in situ without involving movements to and from memory.

Standard memory protection operations apply to I/O registers. This system, plus the UNIX integration of file and device handling radically improves performance.

Linking the various devices is a buffered I/O bus which acts as the interface between the central processors and the memory/video system, Input/Output Processor (IOP), Rasterop hardware, bootstrap ROM, non-volatile RAM, Interrupt Controller, real-time clock, expansion ports and I/O device controllers.

Generalised I/O operations cover control of the Mouse and keyboard input devices, the display ouput device, which has already been described, and the memory-based I/O devices such as disk drives and tape streamers. Keyboard and mouse events are handled by a dedicated 8-bit IOP which communicates with the CPU through a 128 byte dual-port RAM integrated into the IOP chip. A major responsibility of the IOP is the translation of mouse movement into cursor movement without the uneven and unpredictable response that many other UNIX systems provide. The CPU monitors the current position and advises the IOP of the location of windows which might have some bearing on cursor behaviour.

Window handling often involves changes to the cursor. Up to four cursor maps are stored in memory, and the appropriate set of cursors is selected by the CPU. The cursor logic system is controlled by the IOP chip, and is responsible for the control of the major cursor attributes, such

as the Cursor Number (0 to 3), the (x,y) coordinates of the cursor within the display image, and the position of the cursor's "hot spot" pixel relative to the cursor origin. These attributes are used to control the mixing of the cursor image with the screen image to provide an overlay of the two at the appropriate screen position. Mixing may occur according to OR or XOR logic.

The IOP uses the 128 byte RAM to keep a record of the cursor's movements, so that application programs can see exactly where the mouse has been. In addition, the IOP contains code to provide a simple ASCII keyboard facility, although the CPU under software control can override the IOP's existing Keycode-to-ASCII translation procedures.

The handling of memory-based I/O devices such as disk drives and tape streamer is based on a centralised Direct Memory Access (DMA) controller which is interfaced to the system bus and performs fast data transfer operations. In addition, any DMA devices on the expansion port are handled by this controller.

The Ethernet port is provided with an integrated network controller which handles the IEEE 802.3 protocol and provides its own DMA services.

Devices attached to the IBM PC compatible expansion port are serviced by a bus adaptor which performs signal timing and bus-width conversions, and provides buffering facilities.

3.2.2. The Monitor.

The MG-1 display unit is a high resolution (1024×800 pixels) landscape format (ie width > height) 17" monitor using a "white phosphor" screen. The viewing angle is adjustable by way of the knobs at the side of the screen which raise or lower the foot at the back. This feature, plus the two metre cable attaching the display to the computer unit, means that the monitor can be adjusted to suit the individual user. The brightness control is in the lower right-hand corner of the display unit.

The system is supported by a Display Processor, which in conjunction with a memory mapping system supplies rapid image refreshing from any part of the MG-1's memory. This direct memory linkage, plus the high image-refreshing rate, gives a high quality, flicker-free image. For further details of the MG-1's display-handling system, see the "Processors" section, above.

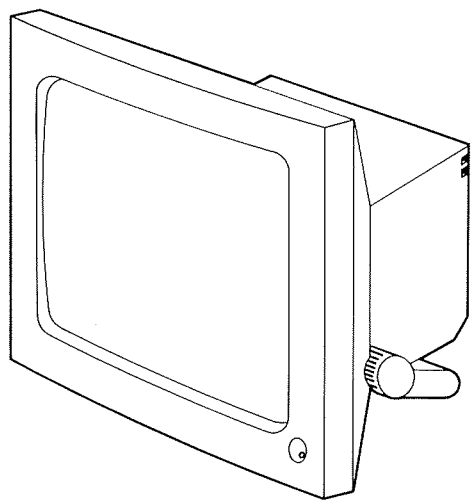


Figure 3.3. The Monitor.

3.2.3. The Keyboard.

The MG-1 keyboard, like the display unit, is of an ergonomically high standard, comprising an adjustable-angle unit attached to the computer unit by a three metre cable. The typing angle is adjusted by way of the snap-lock feet at the rear of the unit.

The cable has two plugs: the first connects the keyboard to the computer unit. The second is used as a Mouse signal relay when the Mouse is attached to the keyboard instead of the computer unit.

The keyboard has a high quality IBM PC layout, and consists of the Typewriter section, with the standard QWERTY pattern, the ten programmable Function Keys, and the Numeric Pad, a total

of 83 stations, all offering auto-repeat. All of the keys may be assigned character or command sequences; full details are given in section 5.8.4.

3.2.4. The Mouse.

The Mouse is a graphical input device consisting of a roller and a set of three buttons. Menu options or points on the screen (for example, windows) are selected by rolling the Mouse across a flat, preferably non-shiny, surface. The rotation of the ball is encoded, and the displacement transferred to the screen via the I/O processor of the computer unit. Commands are relayed via the three Mouse buttons. The unit is attached to the keyboard or directly to the computer unit. If the former arrangement is selected, both of the keyboard cable plugs must be used. The second is the Mouse signal relay.

To use the Mouse, the unit is held in the palm of the hand with the roller on the work surface. The middle three fingers are placed over the Mouse keys.

3.2.5. The RS-232 C Port.

The serial port is used to connect terminals, printers and a number of other devices to the MG-1. Like all MG-1 I/O peripherals, it is driven through the /dev directory, in this case /dev/ttys0. The operating system supports a full range of flow control signals for device sensing and transmission control. For full details, see the 'sio', 'tty' and 'newtty' references in Volume I of the GENIX Programmer's Manual.

In order to increase the number and variety of peripheral devices, the 'acc' facility can be used to drive a two port asynchronous serial board through the IBM bus. The pin allocations for the serial devices are given in Appendix B.

3.3. The Software.

The MG-1's operating system is a National Semiconductor implementation of AT&T's Bell Laboratories System III UNIX, incorporating Berkeley 4.1bsd enhancements. Further details of the UNIX environment are given in Chapters 6 and 7, and full details of the commands available are given in the GENIX Programmer's Manual. The graphics and window handling software are described in this manual, especially in Chapter 8, and in the documentation listed in Appendix F.

All UNIX-derived operating systems comprise three basic elements, the kernel, the utilities, and the shell.

3.3.1. The Shell.

The user passes commands to the kernel and utilities through the shell, of which there are two provided, the C Shell and the Bourne Shell. Although there are differences in such areas as the syntax of commands, system messages, and the symbol used as a system prompt, they operate in much the same way and it is possible to switch from one to another.

The shell is the interface between the user and the facilities of the operating system, and is almost exclusively a mechanism for interpreting commands that invoke other programs. Although the shell has a few built-in facilities, almost all of the shell's job is to activate programs external to the shell itself. In addition to being a command interpreter, the shell is a programming language in its own right, as conditional loops and other control-flow primitives are available. In common with many programming languages, the shell allows the combination of commands, the use of string-matching metacharacters such as "wild cards", and the use of string-valued variables. Combinations of commands may be held in executable text files.

3.3.2. The Utilities.

The first internal layer accessed by the shell is the set of utilities. These include programming languages, text editors and formatters, communications packages, and graphics systems, for example:

1. The C compiler. C is the high level language in which most of UNIX is written. It is accompanied by a large library of standardised functions such as I/O routines.
2. A number of software development tools are supplied, such as the LINT program checker for the C language, which traps syntax errors, portability problems, data type mismatches, and other likely errors; and the M4 Macro Processor which acts as a front end to functionally based languages such as C and FORTRAN.
3. The YACC system ("Yet Another Compiler-Compiler") is a system that will generate part of a compiler, using a user-supplied description of the language system.
4. The SCCS Source Code Control System records versions of source code produced during project development. For example, old versions of a program can be retrieved after editing. SCCS also controls source code files where more than one programmer is working on a system so as to prevent the same file being amended by two programmers at once.
5. The MAKE software maintenance system ensures that files requiring the update or availability of other files can be generated automatically. A full system of inter-file dependencies can be analysed with minimum effort for the programmer.
6. The NROFF and TROFF high quality document preparation systems format text for typewriter-type terminals and phototypesetting systems respectively. Both accept lines of text interspersed with lines of formatting commands which may control layout, heading and footing, pagination, paragraph labelling, font and pitch control, and special character facilities. A macro facility allows the definition of command sequences using single identifiers.
7. A number of text processors are provided, such as the EDIT and EX line editors, the VI screen editor, the SED stream editor, the TBL and EQN tabulated and mathematical text formatters, and the REFER bibliographical system.
8. The DC interactive calculator is available for integer arithmetic. It may be programmed using the BC compiler.
9. The GREP and AWK systems search for patterns in data, and initiate sequences of other commands if the relevent patterns are found.

All of these utilities are described in greater detail in the GENIX Programmer's Manual. There are a large number of optional GENIX third party software packages available. These are easily implemented and include their own documentation.

3.3.3. The Kernel.

Both the shell and the utilities depend on routines provided by the kernel.

The kernel communicates with the shell and the utilities through only 70 or so "system calls" or entry points. Conventional operating systems running at the UNIX level of power often rely on hundreds of such entry points, each controlled by masses of protocol, so the fewer the system calls, the more efficient the operating system. These system calls service resource requests from both the user's program and the utilities, and in fact, UNIX does not distinguish between the two.

One implication of the small number of system calls is portability; inter-system compatibility is determined by the system calls, and so UNIX programs are unusually portable.

3.3.4. The File System.

The kernel is responsible for all I/O device-handling, memory map control, file and directory handling, and program execution. The kernel is based around a hierarchical, tree-structured file system. Files are grouped together in directories of successive levels, and defined by a "pathway" from the "root" directory. This file system treats I/O devices in the same way as files, and groups the device handling routines in a directory called /dev. User files are in a directory called /users, and may be sub-divided into grouped directories of lower levels. Public commands are held in a number of directories such as /bin. Many of the file system elements are outlined in the 'hier' reference of section 7 of Volume I of the GENIX Programmer's manual.

This file structure is the basis of the entire operating system; the kernel is largely concerned with implementing it, and the shell and utilities derive much of their power from it.

Because the MG-1's operating system is so well structured, yet variable in its resources, it is possible to enhance the software with a number of powerful systems. Accordingly, Whitechapel Computer Works have produced the Window Manager, and implemented a powerful graphics handling system. These are outlined in the following sections.

3.4. The Window Manager.

The Window Manager is a basic part of the Whitechapel Computer Works' program of UNIX enhancement, and lies at the heart of the MG-1 programming environment. The display screen is handled as a series of rectangular windows.

The idea of multiple screen windows allows simultaneous use of several data sources, for example in the concurrent display and control of multiple programs, or the examination of different areas of a database or spreadsheet. This operates through the impression that each window is a separate screen in its own right.

The Window Manager creates and manipulates windows on the basis that the user should have full control over all existing windows while affecting the operating system kernel and any active applications as little as possible. To this end, the Window Manager is based on the following principles:

1. All existing windows must be active. Accordingly full access and operational control is available even when a window is partially or wholly obscured by other windows. Keyboard input is directed to a window selected by the user.
2. Each window has access to the full set of user interface tools for scrolling, window subdivision, and screen movement.
3. To avoid conflict with window based applications, the Window Manager instruction set is as neutral as possible. For example, the mouse does not distinguish between its buttons for many commands so as to avoid overlapping with application button-assignments.
4. The kernel is minimally involved. While it retains control over bit-map operations, these relate to the lowest level handling of window primitives. Higher level routines such as the window user interface are handled by the Window Manager. This arrangement has the advantage of overcoming the problem of kernel scheduling delays.
5. Full interactive techniques are available to the application as the Window Manager does not interfere with the standard range of MG-1 operations.
6. The system meets the requirements of programmers as well as application users. These tend to be less well defined, and require support from the full array of operating system development tools.

Windows allow the use of multiple screen displays. As a screen management device, a window may be kept in its active form on the screen or stowed as "icons". Icons are used to indicate the status of windows currently in existance, but not necessarily in use.

All window controls operate through the mouse: windows may be enlarged or reduced, moved, sub-divided, and assigned a lower or higher priority. Mouse button assignments are neutral so as to avoid clashing with application-based controls.

Mouse movements are tracked across the screen by the cursor. Changes in the cursor reflect the presence of different regions within the windows. Four 64×64 pixel cursor rasters are available. The cursor may be trapped by specific processes when particular commands require uninterrupted input.

Keyboard input is directed at the window currently assigned top priority. Such windows are called "listeners", and only one may be active at a time. The Window Manager includes a full TTY Emulator to emulate a standard UNIX character stream terminal.

Three types of window are available. "Full-function windows" operate through rapidly updated bitmaps representing the data objects wholly or partially displayed in the window. "tty windows" are updated from a character map rather than a bitmap, and are designed to handle applications which require a standard keyboard terminal. This system is based on a "tty emulator" where "tty" is a rather out-dated term standing for "teletype". "Physical panel windows" are designed for use in highly interactive applications where high-speed display updates are required. The application writes directly onto the memory holding the bitmap. These windows are of full screen width, and

may not be moved horizontally or reduced in priority.

Overall, the WCW Window Manager contains the kernel software appropriate to handling the most primitive window elements; the Window Manager control system which translates these primitives into user functions; a recommended instruction set to drive the software, which may be user-redefined; and a full set of documentation.

3.5. MG-1 Graphics.

The MG-1 is designed to operate as a high-standard graphics workstation as well as a generalised data processing system. Special image-handling functions are provided by three basic elements, the MG-1 Graphics Library, the Rasterops control system, and the various high level graphics handling packages.

Traditional interactive graphics systems rely on three components, a "frame buffer" or memory area to hold the digital representation of the image; a series of display-control devices which act as the interface between the memory and the screen image; and a monitor. The frame buffer is supplied from the main memory, and because it is an intermediate device, it often causes a significant reduction in performance.

The MG-1 relies on a different and more efficient system. The image representation is supplied directly from main memory, and the full range of graphics operations is thereby increased in speed. Image representations are held in non-contiguous pages of Virtual Memory, and are stored and mapped in exactly the same way as any other data object.

The heart of the graphics-VM combination is the Mapping RAM unit which holds the addresses of the virtual pages and uses these to translate the logical addresses into physical addresses. This means that graphics handling uses existing circuitry, and is fully integrated into the Memory Management Unit. Because memory is used by both 'standard' and graphics applications, memory control circuitry is provided so as to arbitrate between system access to memory and video "refreshing".

Video refreshing is the action of reading bits from memory and converting them to a video signal to compensate for the image fading and to allow changes in the picture to be displayed smoothly. The screen is refreshed 57 times per second to prevent image flicker. Video refreshing has priority over CPU access but the two are so timed as to prevent noticable interference.

3.5.1. Rasterops

The rasterop hardware is at the heart of the MG-1's powerful graphics capabilities. It consists of a number of devices that enable logical operations to be performed between "rasters". A raster is a memory representation of a rectangular array of pixels where each of the pixels within the raster is either off (for white) or on (for black). In the simplest terms, pictures are built up by switching pixels to 'on'.

Rasterops are logical operations performed between rectangles within rasters, where each rectangle is defined by the cartesian coordinates of the displacement from the raster's origin. This system allows the identification of a rectangle in (x, y) terms irrespective of the raster's position on the screen.

Images are created by way of the graphics library or by manual pixel setting. By a process known as scan conversion, the pixels to be set 'on' and 'off' are translated into raster form. Scan conversion is an automatic procedure within the general set of raster handling routines.

Rasters may be manipulated by way of the 16 binary boolean logical functions to combine or duplicate graphics objects, or to simulate image movement. These are listed in section 8.1.3.3. Raster handling is obtained by using one or more of these sixteen available operations. A rasterop involves two rasters, called "source" and "destination", combines these by way of some function, and places the result in "destination". A simple rasterop is to combine the contents of two rasters.

Various techniques are available to cope with differences in the sizes of the source and destination rasters. For example, where the source is smaller than the destination, the source is vertically and horizontally copied to fill the destination, a process called "tiling". Where the source is the larger of the two, "clipping" displays only the visible segment.

Operations are performed on the corresponding bits of source and destination rasters. It is therefore an easy matter to change a raster as well as moving it around the screen, simply by altering the intensity value of particular pixels in memory.

3.5.2. The Graphics Library.

The Graphics Library provides "primitives" for the creation of graphical output such as points, lines, arcs, circles and text characters, in raster form. Area filling is implemented to produce solid black or white shapes.

The graphics library is designed to provide a set of fundamental operations for the higher level graphics packages. Graphics library functions satisfy a number of basic requirements of a graphics handling system, such as the naming and identification of rasters, and the production of primitives while handling such problems as identifying "brush positions", and image clipping.

All primitives are drawn from an initial brush position defined by the start pixel of the operation, and end at a terminal brush position. Clipping occurs when the source image is too large to fit out of the destination raster. The image is analysed for the identity and extent of its visible portion, and the rest is clipped into the destination raster. The terminal brush position is retained, however, as if no clipping had occured.

A second operation is the provision of arguments to the drawing function, so that the primitive being drawn can be combined with the pixels of an existing destination raster.

3.5.3. High Level Graphics Packages.

The MG-1's graphics systems provide a support environment for a variety of optional high level graphics packages. These graphics systems provide routines written in such languages as FORTRAN, Pascal and C to handle device-specific I/O hardware where the objective is the provision of device-independent graphics facilities. Because the basis of any high level graphics package is the existing graphics control hardware, MG-1 supported systems are based on raster imaging.

There are three general areas of interactive graphics handling, namely output, user-requested input, and system management.

Output revolves around a number of standard graphics types and, often, an additional facility for the creation of non-standardised objects. Images may consist of connected lines or rectangles defined by arrays of points, or of closed polygons, or filled areas. Text may be handled by way of fonts that define the size, shape and identity of the characters. By combining objects and text, annotated diagrams or graphs may be created. A further facility may be a generalised drawing facility which produces non-standard and device dependent output. The objective of this last facility is the provision of non-standardised functions in a standardised manner. While not wasting the individual strengths of a specific hardware system, such high level graphics systems retain their predictibility.

Groups of output primitives may be combined to form "segments" which may be manipulated as a whole by way of an identifier. Segment manipulation covers image movement, size changes, rotation, combination by way of Boolean operations, and visibility control. The segment approach allows a small number of primitives to provide the basis of a whole screen. Output segments may be stored on disk in sequential files.

These high level graphics systems may be combined with the Graphics Library and the raster handling routines, to provide a comprehensive range of graphics handling facilities.

Output may often relate either to the full screen or to individual windows. Window control is based on a series of transformations that map data onto physical areas of the screen. These transformations may be user-defined, except where the control of full size screen mapping is concerned.

Input routines operate on explicit user-requests. Data sent to the system may include coordinates relating to a point or line, a single real value, a segment identifier, an integer relating to selection of a menu option, for example, or a text string entered from the keyboard.

Workstation control routines are also provided, to include display and buffer up-dating, form feeding for plotters, segment drawing from currently defined data, and message handling. An error-handling system should return a full range of error messages.

3.6. Options.

The MG-1 supports a wide range of optional features. These are either standard devices or systems which which have more than one possible configuration, or features which may be omitted completely without compromising the performance of the overall system. Options will include memory expansion, a tape streamer, different hard disk capacities, the Ethernet networking system, the IBM PC expansion cards, and the colour monitor.

3.6.1. Memory Expansion.

The standard MG-1 main memory configuration is 512 Kbytes. However, the computer unit will accept up to seven 512 Kbyte expansion cards, or three 2 Mbyte cards, raising the total to either 4 or 8 Mbytes.

3.6.2. Hard Disk Configurations.

The hard disk options begin at 10 Mbytes. This configuration does not include the on-line programmer's documentation, as the full version amounts to a total of 1.4 Mbytes. Documentation is available on the higher capacity options, in progressive degrees of completeness. In order to free disk space, documentation, games and other files may be deleted.

3.6.3. Ethernet.

Ethernet is a local area network (LAN) system which allows up to 200 MG-1 workstations to be connected together by way of 50 ohm cables of up to 500 metres length. If repeaters are used, more than one 500 metre length can be used between terminals. Data transmission occurs at 10 Mbits/sec.

The IEEE 802.3 Ethernet communications port performs the link-level protocol functions and inter-host network accessing together with memory management, error reporting, packet handling and processor interface functions.

3.6.4. Peripheral Expansion.

The computer unit's general purpose expansion port provides direct access to the buffered processor bus and the centralised Direct Memory Access (DMA) controller. By adding the IBM PC motherboard, up to three expansion cards may be fitted. This board provides full emulation of the IBM PC bus.

Data is translated from the MG-1's 16 bit word to the IBM PC's 8-bit word by using two consecutive IBM words.

The 1 Mbyte IBM PC address space (strictly speaking 1 Mbyte minus 128 Kbytes) maps onto MG-1 address space as follows:

IBM PC	0x00000 – 0xDFFFF
MG-1	0xD00000 – 0xDDFFFF

The 64 Kbytes of IBM PC I/O space maps onto MG-1 address space as follows:

IBM PC	0x0000 – 0xFFFF
MG-1	0xDE0000 – 0xDEFFFF

The interrupt lines from the IBM PC expansion bus may be configured to drive a choice of five interrupt inputs on the MG-1 interrupt controller.

Chapter 4
Installation

4.1. Setting up the MG-1.

The MG-1 computer unit should be located within 2.5 metres of a 13 AMP standard national grid voltage power point.

The mounting surface should be clear of loose debris and adequate ventilation should be available around the unit. In particular, do not cover the fan outlet at the rear of the unit.

When work space is limited, the computer unit can rest on its side or under the desk, or else the monitor can rest on the computer unit.

Inspect all plugs and connectors to ensure that none of the pins are broken or bent.

Begin by ensuring that the mains power supply is switched OFF.

Connect the female plug on the mains supply lead to the socket on the left hand side of the computer unit's rear panel. Do not plug in the power supply at this point. Powering up the computer will be covered in the next chapter.

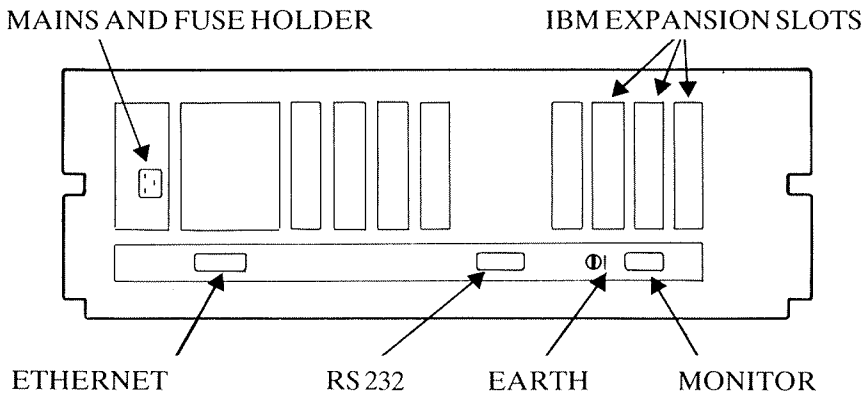


Figure 4.1. Rear Panel Connections.

Connect the monitor's flying lead to the D socket on the computer unit rear panel marked "monitor". Slide the earthing clip of the monitor cable onto the spade terminal located on the left hand side of the monitor socket.

The knobs at the side of the monitor are used to adjust the viewing angle of the screen, which may be tilted backwards from the vertical by upto 17°.

Connect the keyboard to the computer unit by attaching the shorter of the two leads to the D socket marked "keyboard" beneath the front flap of the computer unit.

When the mouse is to be connected to the keyboard rather than directly to the computer unit, the longer of the two keyboard leads is plugged into the D socket marked "mouse". This arrangement increases the area over which the mouse may be used.

If the mouse is to be attached directly to the computer unit, its lead should be plugged into the "mouse" D socket, and the longer of the two keyboard leads is not used.

The keyboard has adjustable feet for altering the typing angle. Two positions are available, horizontal and inclined. Applying outward pressure to the keyboard feet causes them to click into their extended position which inclines the keyboard.

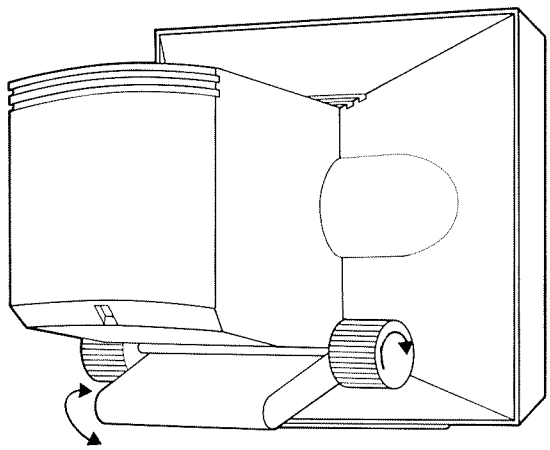


Figure 4.2. Viewing Angle Adjustors.

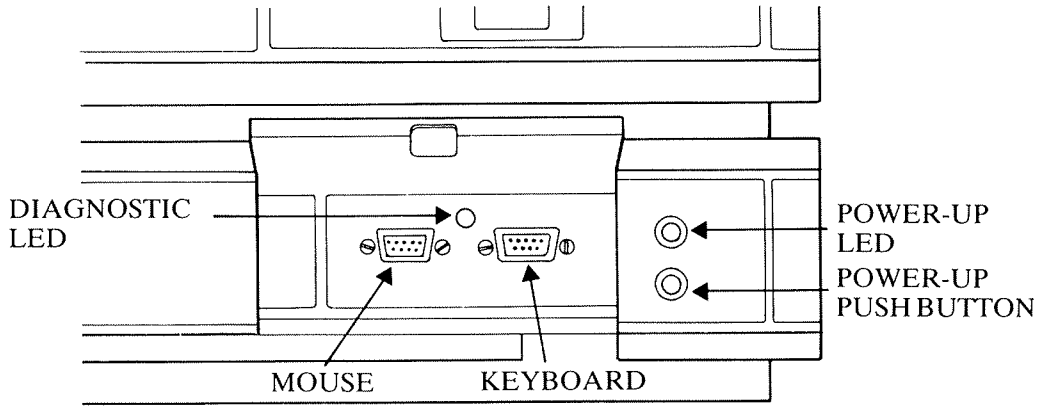


Figure 4.3. The Keyboard Lead.

Note that the mouse performs best when used on a surface that is not shiny or polished. This is because enough traction must be available to translate each Mouse movement into rotation of the ball bearing under the mouse.

4.2. Floppy Disks.

The MG-1's floppy disk drive uses standard 5.25 inch, double-sided, double-density, soft-sectored, 96 tracks per inch (tpi) disks. Because disks are manufactured in standard form, they must be "formatted" so that data structures specific to the MG-1 can be stored and accessed. The formatting procedure is described in section 9.14.

Floppy disks are quite delicate and care should be taken to preserve them in order to avoid the loss of valuable data. The following precautions should be taken:

Handle floppy disks in their protective jackets.

Do not expose them to electromagnetic fields such as generated by telephones or other electrical appliances. Even the MG-1 units may be capable of affecting disks, so do not place them on the equipment.

Avoid exposing disks to direct sunlight or moisture.

Do not use paper clips on disks and use only felt-tipped pens for writing disk labels once they have been affixed.

Do not touch the magnetic surface of the disks, and take care not to allow debris to come into contact with the surface. A single scratch will ruin a disk.

Do not bend or fold disks, and take care to provide protective wrapping when mailing disks.

4.3. Installing the Bus Adaptor.

To install the IBM PC bus adaptor, carry out the following instructions:

1. Ensure that the power is OFF. For power-down routines under software control, see section 5.10.
2. Remove the cover from the MG-1 computer unit by releasing the holding screws (located underneath and to the rear of each side of the unit) and sliding the cover forward.
3. Remove (and retain) the fixing screws on the PC Bus supports located approximately one inch in front of the left hand side of the monitor socket and ethernet socket.
4. Screw into each of the two PC Bus supports one of the adaptor supports supplied with the adaptor unit.
5. Gently press the 64 pin DIN41612 male socket on the adaptor unit into the 64 pin DIN4162 female socket located approximately seven inches in front of the monitor and ethernet sockets and running parallel to the front panel of the computer unit.
6. Secure the adaptor unit by screwing into the adaptor supports the two fixing screws removed in step 4 above.

4.4. Installing Expansion Cards.

To install an IBM PC bus expansion card:

1. Ensure that the power is OFF. See section 5.10 for power-down routines.
2. Choose any of the available expansion slots on the adaptor unit.
3. Remove the blanking plate on the back panel of the computer unit opposite the slot chosen for the expansion card.
4. Snap the card guide supplied with the expansion card into the holes on the vertical support panel opposite the slot.
5. Firmly press the expansion card into the slot.
6. Replace the screw, removed in step 3, to secure the bracket and card.

If the card being installed has configuration options these will be described in the documentation supplied with the expansion card.

Chapter 5
Getting Started

5.1. Powering Up.

Powering up the MG-1 requires two actions from the user:

1 Turn mains power ON

2 Press front panel ON button

There is a short delay, while diagnostic routines check the status of the system and the bootstrapping procedure executes. The diagnostic LED illuminates briefly and the hard disk begins to run up to full operational speed. The role of this diagnostic LED is described in section 11.2. After a delay of about 15 seconds, the screen displays a sequence of regular memory diagnostic patterns which indicate the status of the MG-1's memory. These continue for about 10 seconds. The screen clears to white and the following message is displayed:

Whitechapel Computer Works MG-1 (dd-mm-yyyy)

The date indicates when the ROM software was issued.

As soon as the disk drive reaches full operational speed, the MG-1 bootstraps the operating system from track zero of the disk. During the bootstrap procedure the MG-1 displays the following message together with an indication of system size:

Loading from hard disk
Boot: hd(0,0)vmunix

After 15 seconds or so, the screen clears and over the next two minutes the system displays the operating system identifier, and details of the system configuration, administration and initialisation, then the operating system banner followed by the "login" prompt.

The first time a GENIX configured MG-1 is powered up, the user can log in either as "root" or as "guest". To log in as root the user is advised to be thoroughly conversant with system administration (see Chapter 9) and system security (see Chapter 10) and to create a personal entry in the password file for subsequent logins when the wide-ranging powers of root are not required.

To log in as "guest", for the purposes of gaining initial hands-on experience, type "guest" and press the <RETURN> key (labelled with a clockwise arrow). The user new to GENIX should now read Chapter 6 and then work through the example session in Chapter 7.

For powering down at the end of a session, see section 5.10.

5.2. Bootstrap Options.

The MG-1 system ROM contains control routines which automatically load the operating system from the hard disk into the main memory. This operation is known as "bootstrapping". The ROM also identifies problems that might prevent successful bootstrapping, for example memory hardware faults.

In addition to the control and diagnostic routines provided by the ROM, various systems are provided by the floppy disk supplied with the MG-1. The start-up routines include a bootstrap menu which allows the user to specify the sequence of diagnostic routines activated, including running the facilities on the floppy disk.

The default sequence of events involves the MG-1 booting up from hard disk hd0 but various power-up options are available and are selected by interrupting the normal bootstrap process. If the MG-1 detects any character from the keyboard, or a <Ctrl><a> character from the RS-232 serial port, at any moment between the power ON button being pressed and the operating system identifier appearing on the screen, the bootstrap options menu is displayed.

Choose from one of the following:

0 – Boot from hard disk 0
1 – Boot from hard disk 1
2 – Boot from floppy disk
3 – Enter monitor
4 – Switch off

5.3. Loading from Hard Disk.

To load from a file other than the default file which is hd(0,0)vmunix, select the first menu option by pressing numeric key '0'.

Then type the device name, the drive number followed by the partition number, and the pathname of the file to be loaded.

The syntax of the response is:

<device>(<drive>,<partition>)<pathname>

Examples:

hd(0,0)vmunix
fd(0,0)vmunix
hd(1,0)vm.new

In the event of any of the files specified in this way containing errors, there is a full range of error messages available.

In the event of the required file being stored on a second hard disk, option '1' from the menu should be selected. The user response is as outlined above.

5.4. Loading from Floppy Disk.

To load from a floppy disk, such as the diagnostic package supplied with the MG-1, insert the disk in the drive, close the drive door and select option 2 from the bootstrap menu.

5.5. Monitor.

The WCW ROM debugger is a sophisticated monitor program that allows examination and patching of memory (both system and I/O) and processor registers. The range of options provided should offer an adequate medical kit for the repair of memory-based problems.

To enter the ROM debugging monitor select option 3 of the menu. The "WCW Monitor" banner appears on the screen and the system awaits user commands specifying memory areas for examination or modification.

5.6. The fsck System Checker.

The diagnostic routines activated during bootstrapping check on the status of the hardware and its support software. As a further guide to system integrity, GENIX automatically runs the 'fsck' system. This activates an 'interactive file system consistency check' which determines whether a variety of file attributes are fully consistent. For example, disk blocks may apparently be owned by more than one file at once, or not accounted for at all, or else disk space reports may not tally.

In such cases, the fsck system proceeds to correct any problems, but requests user permission to proceed in each case. This is because almost all such problems involve the loss of some data. A list of all such data losses is given.

Where no errors are detected, the fsck utility reports on the number of files on the disk, and the number of blocks free and currently in use.

More information on the fsck command is available in Chapter 9 and in section 8 of Volume I of the GENIX Programmer's Manual.

5.7. The stty command.

The 'stty' command sets I/O options on a current output terminal. A wide range of options may be set, such as flow control, parity settings, echo on output characters, procedure killing facilities, and character case mapping.

If no arguments are specified, the command produces a report on terminal speed (baud rate) and those options whose current settings are different from their default values. A number of reporting options are available, and the full range of stty options is very large.

The stty command is covered in more detail in section 1 of Volume I of the GENIX Programmer's Manual.

5.8. Using the Keyboard.

The MG-1 keyboard is an input device which can be configured by the GENIX operating system to specific user requirements. This section describes the standard default configuration. Certain software systems may alter the keyboard: refer to the package documentation for application-specific configurations.

The MG-1 keyboard is divided into three sections: Typewriter Area, Numeric Keypad (incorporating cursor controls), and Function Keys.

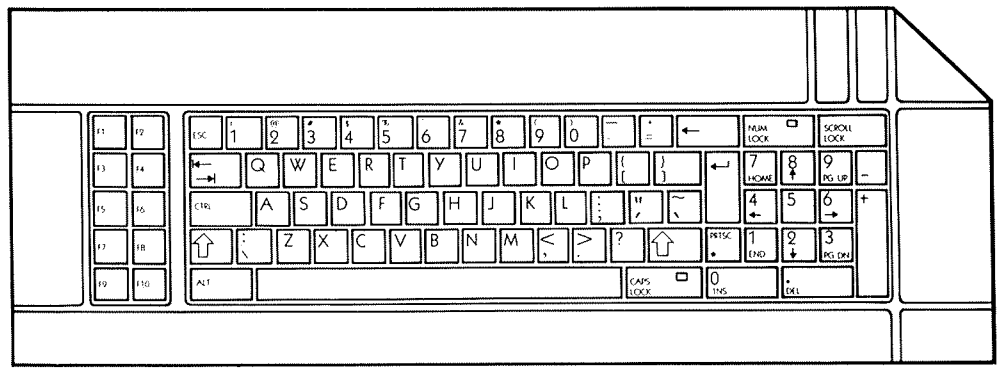


Figure 5.1. The Keyboard.

5.8.1. The Typewriter Area.

Key positions in the typewriter area of the MG-1 keyboard are very similar to a standard QWERTY typewriter. The <spacebar> performs an equivalent function to that on a typewriter.

When depressed, the <Caps Lock> key locks characters A to Z in the uppercase position. Pressing the <Caps Lock> key again releases the uppercase mode for these characters. An LED within the <Caps Lock> key lights up when the key is engaged.

Pressing either of the <Shift> Keys (hollow vertical arrows) shifts the keys in the typewriter area of the keyboard into the uppercase mode. Alphabetic characters are then displayed as capital letters, non-alphabetic characters in the typewriter area are displayed as the character shown in the upper portion of the key.

The <Esc> key is often used in conjunction with other characters to exit from programs, terminate activities, log out, or stop the movement of text on the screen.

The <RETURN> key, marked with a clockwise arrow, is used after typing a GENIX command to indicate that the command is complete and must be executed.

The <backspace> key on the top row of keys, labelled with a back arrow, is used to correct typing errors. The character immediately to the left of the cursor is removed each time the backspace is pressed. Note that all MG-1 keys offer auto-repeat, and that care should be exercised when

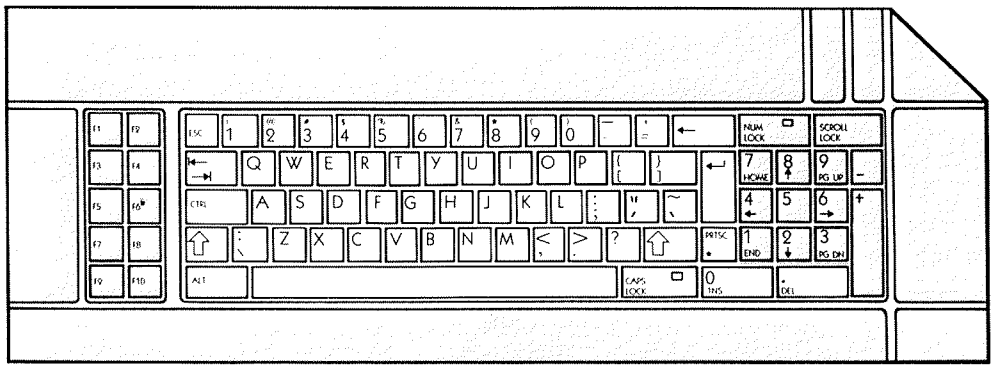


Figure 5.2. The Typewriter Area.

deleting text.

Note that the number '0' and the letter 'O' are not interchangeable, and that the number '1' and the letter 'l' are also completely different.

The function of the remaining keys in the typewriter area are application-specific. Information on these keys operating under specific software control can be obtained from the relevant software documentation.

5.8.2. Numeric Keypad.

Pressing the Numeric Lock Key on the numeric keypad sets keys 0 to 9 to numeric mode. Pressing the Numeric Lock again returns keys 0 to 9 to cursor control. An LED within the <Num Lock> key lights up when the key is engaged.

The function of the <Delete> key is application specific and is defined in the Operating System or Applications Program Manual.

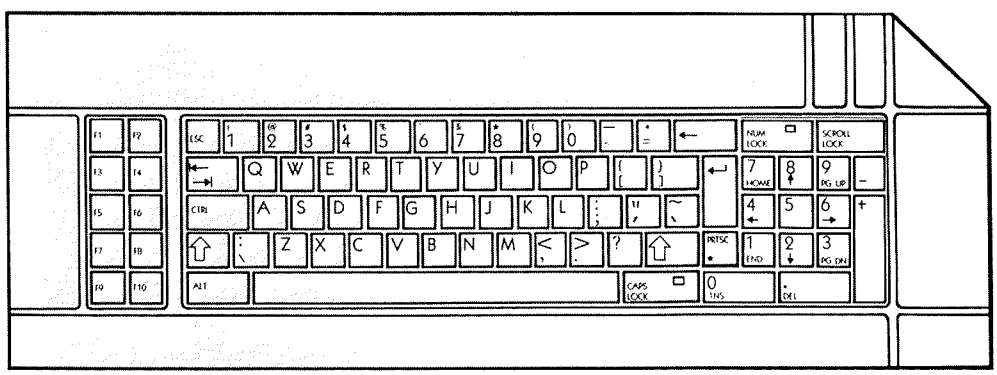


Figure 5.3. The Numeric Keypad.

5.8.3. Cursor Controls.

The Cursor Control Keys may be used by some application programs to move the cursor around the screen: The <Cursor Up> key moves the cursor one line up; <Cursor Down> moves the cursor one line down; <Cursor Right> moves the cursor one character position to the right; and <Cursor Left> moves the cursor one position to the left. All of these keys have an auto-repeat action, and so will provide continuous cursor movement if required.

The <Home> Key moves the cursor to the top left corner of the display.

The <End> Key moves the cursor one position to the right of the last character on the line.

For details of the action of the <Pg Up> (Page Up), <Pg Dn> (Page Down), and <Scroll Lock> keys refer to the Program Manual of the specific software package.

5.8.4. Function Keys.

The ten function keys labelled F1 to F10 are always under program control. Refer to the relevant application Program Manual.

The Window Manager's VT100 emulator allows any of the MG-1's keys to be reprogrammed. The following sequence of commands should be used:

```
ESC [ Pk ; Ps ; Pr ; Pn p
```

where Pk is the scan-code generated by the key, as shown in Appendix D; Ps is the shift state that must be used if the translation is to take place. The value of this parameter can be obtained from the 'include' file <sys/panelevent.h>; Pr is the autorepeat flag, 1 for autorepeat, 0 for non-autorepeat; and Pn is the string to be generated. For example, to set the function key F1 to produce

```
pwd
ls -l
```

the following sequence should be given:

```
% ESC [ 59 ; 0 ; 0 ; "pwd" ; 13 ; "ls -l" ; 13
p
```

where '13' is the ASCII code for a line-feed.

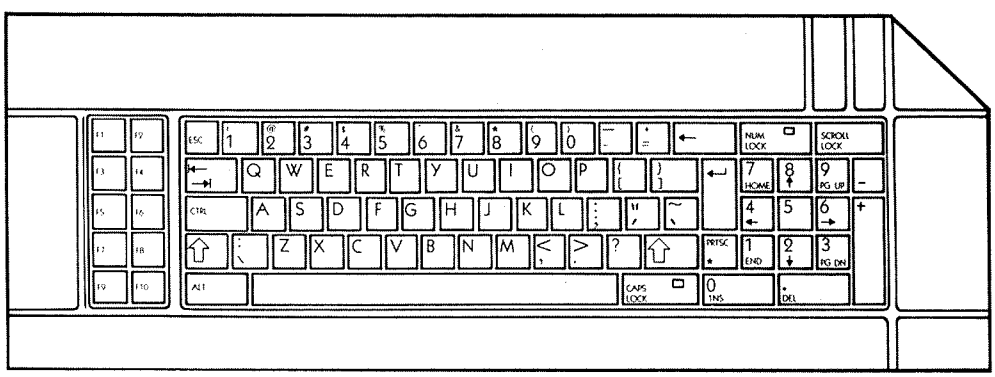


Figure 5.4. The Function Keys.

5.8.5. Control Sequences.

By convention, some keys have a special meaning when using GENIX. These include control keys or sequences used to produce special GENIX characters that do not appear on the keyboard, and various "escape" sequences used to exit from programs, terminate activities, log out, or stop the movement of text ("scrolling") on the screen. Angle brackets (<>) are used in this manual to

indicate particular keys.

When a "control" character is required, always press the <Ctrl> key first and hold it down while the second key in the control sequence is pressed.

The default set of control characters is as follows:

<Ctrl><s>	Pressing the "control" and "s" keys simultaneously will stop text from scrolling.
<Ctrl><q>	Restarts scrolling.
<Ctrl><d>	Has several uses, for example: To log in and log out; To bring the system up from maintenance mode; To produce the End of File character (EOF).
<Ctrl><z>	To suspend a program that has been invoked by the C shell and return to the command prompt.
<Ctrl><y>	Suspend a program that has been invoked by the C shell when it next attempts to read from the keyboard.
<Ctrl><o>	Throwaway terminal output until another <Ctrl><o> is typed.
<Ctrl><x>	Erase a line
<Ctrl><w>	Erase a word

The <DELETE> key is used to abort a foreground progress.

The pipe character (|) is used in some advanced features of GENIX, for example the passing of output from one operation straight into a second operation.

Exceptions to these general rules are detailed for each utility in Volume I of the GENIX Programmer's Manual.

5.8.6. MG-1 Keycodes.

Following the power-up bootstrapping procedures, the MG-1 keyboard is programmed by GENIX to generate DEC VT100 key codes. The correspondence between keys on the MG-1 keyboard and the VT100 keyboard is:

VT100 key	MG-1 key
PF1	F1
PF2	F2
PF3	F3
PF4	F4
. , – ENTER BREAK	n/a
Ctrl SPACE (NUL)	<Ctrl><@>
Ctrl ^ (RS)	<Ctrl><^>
Ctrl ? (US)	<Ctrl><_>
Line Feed	<Ctrl><j>
Tab	<Ctrl>< >
No Scroll	n/a
Setup	n/a

Each MG-1 keystroke is identified by the computer's processors through the medium of ASCII codes. A full list of these is given in Appendix C.

5.9. Setting the Clock.

At the beginning of the MG-1's operational life, the clock can be set to correct the date and time. Against the shell prompt, type:

```
%date yymmddhhmm.ss
```

where the optional 'yymmdd' element sets the last two digits of the year, the month number and the day number. 'hhmm' sets the hour and minutes (on a 24 hour basis), and the optional '.ss' element sets the seconds. If year, month and day are not given, they are set to the current values.

This assumes that they were set when the system was first powered up.

5.10. Powering Down.

Option 4 from the bootstrap menu closes down the MG-1. Otherwise, type "off" in response to the shell prompt.

Chapter 6
GENIX — An Overview

6.1. Introduction.

The GENIX operating system is a derivative of UNIX, developed by AT&T's Bell Laboratories. The MG-1's operating system is the standard System III UNIX incorporating Berkeley 4.1bsd enhancements, implemented on the Series 32000 chip set.

In common with all UNIX systems, GENIX consists of a kernel, a number of utilities, and one or more shells, in this case the C and Bourne Shells. The salient features of these elements have been laid out in the System Description chapter: the present chapter will cover some of these features in more detail, and introduce the use of the shells, utilities, kernel, file system and multiprogramming facilities. The loading of the operating system, and the various bootstrapping options were covered in Chapter 5.

6.2. Portability.

The GENIX user operates the resources of the computer by entering commands through any of the input devices, such as keyboard, mouse or serial RS-232 C port. Although it the kernel that handles the physical resources of the computer, the user interacts with the shell first, the utilities second, and the kernel last.

The shells are command interpreters, which pass user instructions to the lower level elements. The utilities are self-contained packages which provide a set of commands designed to handle certain areas of data processing activity, such as programming, text handling or communications.

The kernel is almost entirely hidden from view. It controls the allocation of CPU time, memory space, and communication channels for the various tasks that system users may have running at any particular time. It consists of a central supervisor and a number of low-level service routines which take care of essential activities such as fetching characters from the keyboard, writing to memory and examining the system clock.

The shell and utilities request services from the kernel through a number of fixed entry points or system calls which act as ordinary subroutine calls. Entry points isolate the utilities from the kernel's internal system, and define a simple interface to the machine hardware.

Versions of UNIX frequently share the same basic set of entry points (system calls) and because it is the system calls that determine compatibility, UNIX utilities and programs tend to be very portable.

6.3. The File System.

The GENIX file system is shaped like an inverted tree. Directory (branch) and file (leaf) names can be up to 14 characters long, and may include any combination of upper and lower case letters and numbers. Do not use the slash (/) character or any punctuation marks except the full stop or period.

Directories are used to group together related files. Each directory may include subdirectories which may include their own lower level subdirectories; there is no limit to the depth of levels within the directory structure.

The branches or leaves directly below the currently active directory are usually the only ones 'in view' but changing to another directory involves a single command.

The user has almost complete freedom as to the structure of the directories.

Figure 6.1 shows part of a file system for two users, Richard (a programmer) and John (a technical writer). Richard has two separate directories for his projects (the .c files contain program source code) and John has a single directory devoted to his manual. John's 'manual' could be a directory containing files that are text sections. Slashes (/) are used to separate the segments of a file name.

Thus the full name or "pathname" of Richard's draw.c file is:

/users/richard/graphics/draw.c.

John's manual directory has the following path:

/users/john/manual

The slash at the beginning tells the system to start the path at the "root". This full name is seldom needed however. The directories called "john" and "richard" are the "home" directories of the two users, and it is within these directories that John and Richard will be positioned when they log in.

From John's home directory he can refer to Chapter 2 of his project as manual/chapt2 (note the absence of the initial slash since this search starts at John's current directory, and not the root). John can change his viewpoint with a 'change directory' command typed against the shell prompt:

```
% cd manual
```

Figures 6.1 and 6.2 indicate the structure of a typical GENIX system.

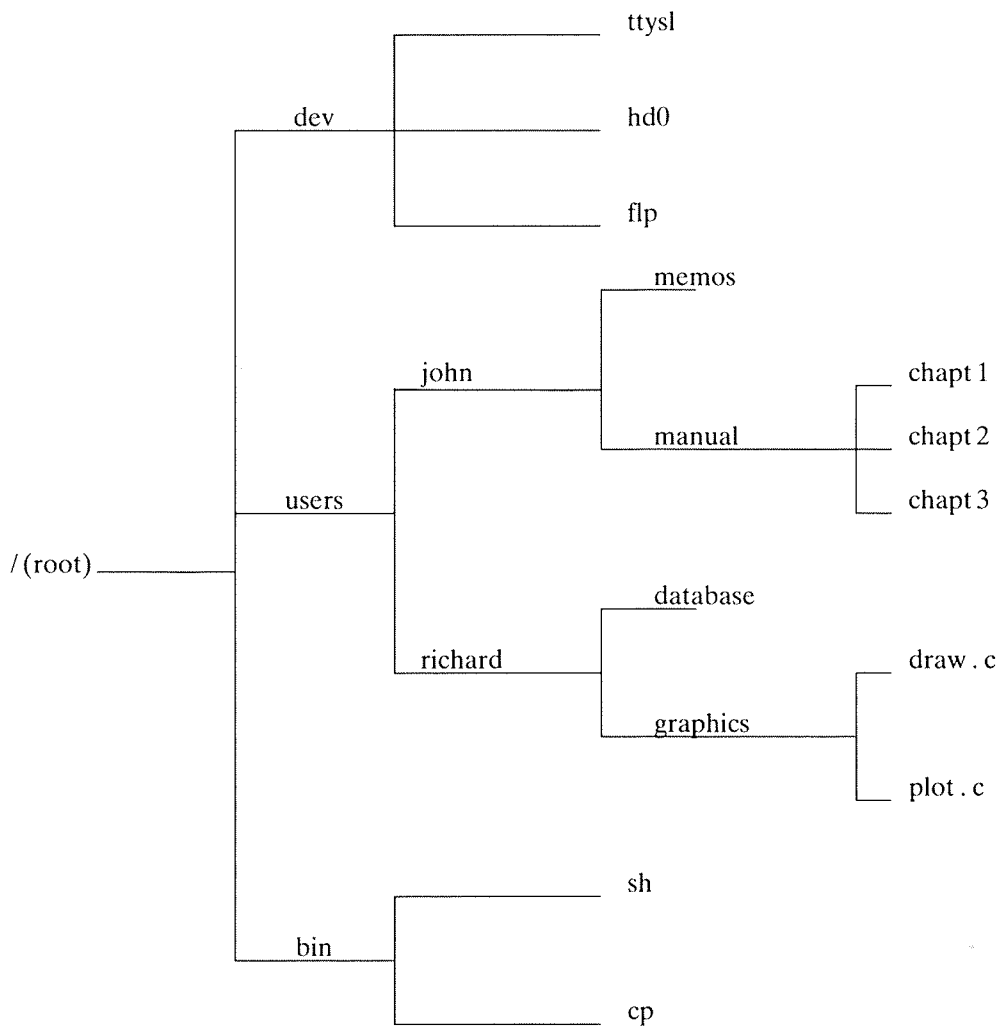


Figure 6.1. Simplified Directory Structure.

A tree structure for files has the advantage that even if a user happens to maintain a large number of files, a small sub-set only need be seen or thought of at any one time.

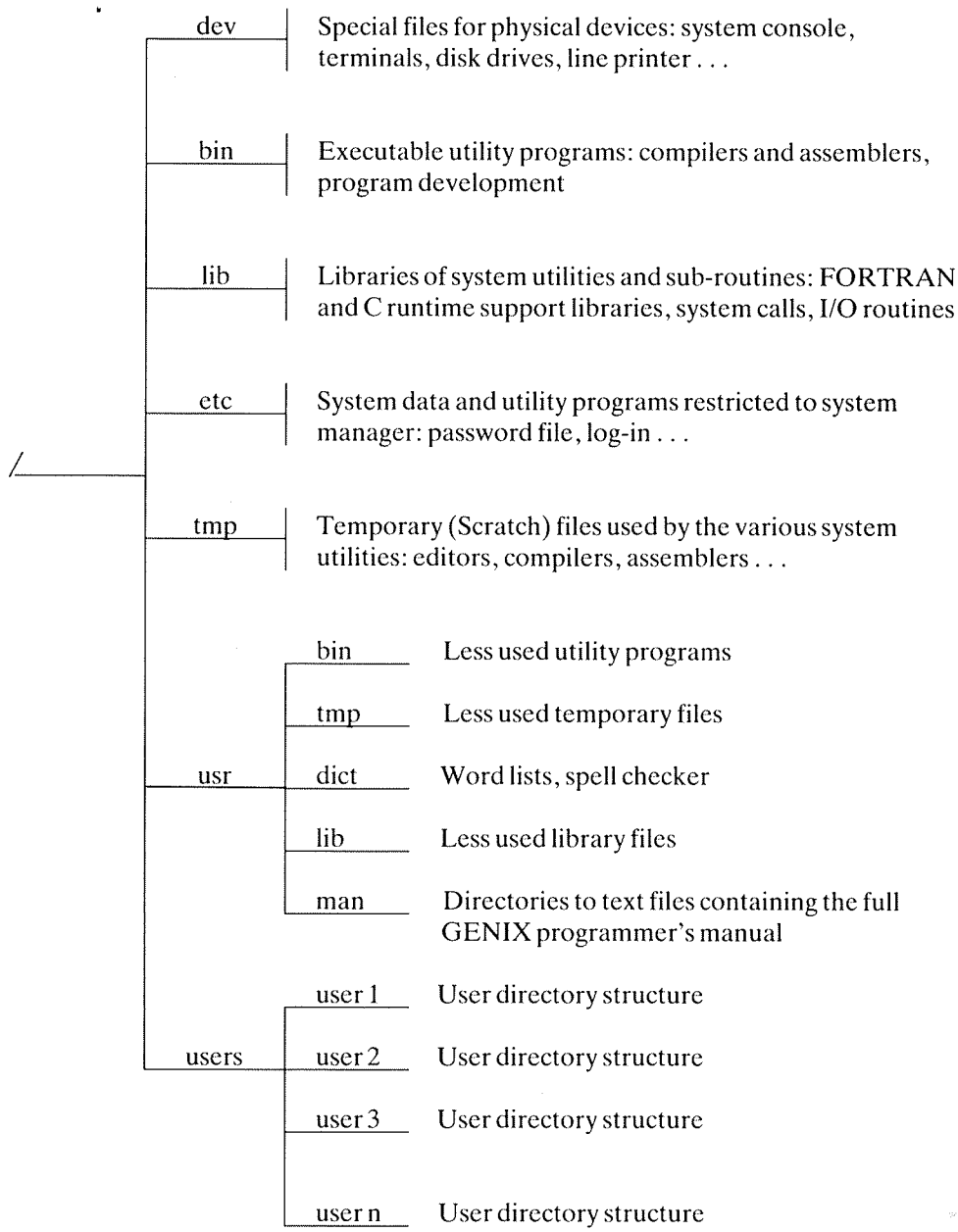


Figure 6.2. Typical Directory Structure.

Two additional directories are shown on Figure 6.1, /dev and /bin. Public commands held in these directories contain the files responsible for the handling of peripheral devices such as the hard disk drives, floppy disk drive, and streamer tape drive. Input/Ouput programming with GENIX is simplified by the availability of files held in the /dev directory which require only normal file read/write instructions for access to peripherals.

Figure 6.2 shows a typical file structure to be found on a GENIX system.

6.4. Selecting a Shell.

The MG-1's default GENIX configuration is the C Shell, whose prompt is the **%** character. The default shell can be changed at any time throughout the user session, including immediately after the login sequence, by typing the following command:

```
% chsh /bin/sh
```

The Bourne Shell is thereafter the one that will be used at the next login.

To change back to the C Shell, type the following command:

```
$ chsh /bin/csh
```

At any point, the shell actually in use can be changed. From the C Shell, type:

```
% sh
```

to change to the Bourne Shell. To change back, type:

```
$ csh
```

The prompt symbol used by either of the shells can be changed during a session by way of the 'set prompt' command within the C Shell or the 'PS1' command within the Bourne Shell:

```
% set prompt = '@'
```

or

```
$ PS1 = '@'
```

Because the above commands use a string declaration to set the new prompt symbol, whole words can be used, for example:

```
% set prompt = 'hello world'
```

would set the shell prompt to the words 'hello world'. Please note that the literal string that will be used as a prompt symbol should be enclosed within " closed quotes" symbols.

6.5. Using the Shell.

When the user types a command into the system, it is stored in a command line buffer until the carriage return character is detected. The command line is then interpreted by the shell for subsequent utility or kernel action.

Each command is a sequence of words separated by spaces or special characters. The first word specifies the command to be executed. Any remaining components, with a few exceptions, are passed as "arguments" to that command.

If the first element of the command names an executable file and refers to a compiled program or shell script, the shell creates a process that executes the command. An executable file is one indicated by an appropriate set of access codes. For further details of access permission codes, see Chapter 10.

If the file is marked as being executable but is not a compiled program it is assumed to be shell script, a file of ordinary text containing command lines. In this case, a new shell process is created which reads the file and executes the commands it contains.

The shell normally searches for simple commands (commands without a slash "/" prefix) in a series of directories in succession, and then runs the first one it finds (if any). This series of directories is

known as the user's "search path". The user may change the shell path variable to include extra directories, exclude directories or change the order of searching.

For example, the C shell command

```
% set path = (. /usr/pg/john/bin /bin /usr/bin)
```

or the Bourne shell command

```
$ PATH = ".:/usr/pg/john/bin:/bin:/usr/bin"
```

will cause the shell to search the current directory ".", the subdirectory "bin" of the users login directory and the two "system" directories /bin and /usr/bin. Frequently used public commands (such as cat, rm, ed) are kept in /bin whilst less frequently used commands live in /usr/bin, /usr/nsc and usr/ucb.

6.6. Generation of Argument Lists.

The arguments to commands are very often filenames. Sometimes these filenames have similar, but not identical names.

To take advantage of this similarity in names, the shell allows users to specify patterns that match the filenames in a directory. If a pattern is matched by one or more filenames in a directory, then those filenames are automatically generated by the shell as arguments to the command.

Most characters in such a pattern are employed at face value, but there are special "metacharacters" that may be included in a pattern. These special characters are the asterisk (*) which matches any string of variable length (including an empty string), the question mark (?) which matches any one character, and any sequence of characters enclosed within square brackets ([and]) which matches any of the enclosed characters.

Inside square brackets a pair of characters separated by a dash (–) matches any character lexically within the range of that pair. Thus "[p–xy]" is equivalent to "[pqrxy]" as well as "[psuxy]".

Examples:

*	matches all names in the current directory
temp	matches all names containing "temp"
[a–f]*	matches all names beginning with "a", "b", "c", "d", "e" or "f"
*.c	matches all names ending in ".c"
/usr/bin/?	matches all single-character names in the /usr/bin directory

This pattern matching facility makes it possible to organise information in large collections of small files that are named in a disciplined way.

6.7. Command Groupings.

Various items of punctuation are used by the shell to separate commands or group them together. Some of these are keywords like "if", "then", "else" and "while" are employed when the shell is being used as a programming language. For a discussion of the shell as a programming language the reader is referred to specialised UNIX text books and to the GENIX Programmer's Manual.

Multiple commands can be grouped on the same line provided they are separated by semi-colons (;). The shell does not prompt for another command until it has executed all of the commands grouped by semi-colons. It is, however, possible to ask the shell to start a program but not wait for its completion before accepting another command. This is achieved by running the program as a "background job", indicated by ending the command with an ampersand (&). To execute a sequence of commands in the background they must be grouped together by parentheses.

In the following example only the 'echo ready' command will be run as a background job after the first two commands have been completed:

```
% date; ls; echo ready&
```

The following command causes the shell to run all three commands as background jobs:

```
% (date; ls; echo ready)&
```

As soon as this command is input, the shell prompt will be displayed, as the three jobs are completed in the background.

6.8. Input/Output.

Many GENIX programs are designed to take characters from a standard input channel, for example, the keyboard, transform them in some manner and write the results to a standard output channel such as the screen. This kind of program is called a filter.

The 'sort' utility, for example, takes a file of text lines from its input, sorts the lines to alphabetical order and writes them to its output. By default, standard output is the screen and standard input is the keyboard. The user can type in lines and see them sorted immediately. More typically the sort utility would be applied to the contents of files.

Suppose 'keywords' is a file containing an unordered list of words selected from this manual to provide the index. The command

```
% sort <keywords
```

instructs GENIX to ensure that the sort program receives its input from the keywords file, sorts its contents and displays the ordered list on the screen. The '<' character, known as a 'redirection operator' is read as 'from' or 'source'.

The command sequence

```
% sort <keywords >index
```

would cause the ordered keyword list to be stored in a file called index and the results would not be displayed. Read the '>' as 'to' or 'target'. The important point about redirection operators is that they are interpreted by the shell not the program. They work with any filter and will connect it to any files or devices on the system.

Therefore

```
% sort < keywords > /dev/lp
```

causes the immediate output of the sorted contents of the keywords file on the line printer, which as far as the shell, and most of GENIX is concerned, is a file in the /dev directory of devices.

Typically, output would not be sent directly to a line printer but rather to an intermediate print spooler, known to GENIX as lpr. The correct way to do this is:

```
% sort <keyword |lpr
```

This tells the shell that output from the sort operation is to be connected to the input of the lpr program. The vertical line is pronounced 'pipe'.

A pipe is a form of redirection in which two or more programs are run together with the output from one being passed as input to the next.

The main advantage of the pipe is that it provides a concise notation for the sort of task that would require the explicit use of temporary files on many other operating systems. None of these intermediate files are required in the course of pipe operations.

The following example uses the document preparation programs described in Volume I of the GENIX Programmer's Manual:

```
% refer myfile >tempfile-1
% tbl <tempfile-1 >tempfile-2
% neqn <tempfile-2 >tempfile-3
% nroff –ms <tempfile-3 >tempfile-4
% lpr <tempfile-4
```

This uses four temporary files which are left in the current directory at the end of the task and is much less compact than its pipeline equivalent.

```
% refer myfile |tbl |neqn |nroff –ms |lpr
```

The result is the same, namely the formatting and printing of a text file while leaving no intermediate files.

Pipelines make it possible to build powerful data-manipulation commands by connecting a series of filters. Moreover, these commands can be incorporated into a "shell script" (a file containing a sequence of shell commands) and invoked as, and when, required.

6.9. Programming the Shell.

The shell is not merely an interactive command language. It is also a programming language in which new commands can be written in terms of existing GENIX commands. Anything which is valid when using the shell interactively from a terminal can be incorporated in a script and usually vice versa.

GENIX supports a very powerful set of tools for use individually or within shell scripts, which are described below. When harnessed together by the shell's control language, these tools can perform major application tasks with the minimum of programming effort. A classic example is the pipeline

```
% ls |grep file |wc –l
```

which prints the number of file names in the current directory containing the string 'file'.

6.9.1. Shell Scripts.

A key feature of shell programming is the shell script, which is an executable file containing a number of other commands.

By way of an example, suppose that hardcopy of various mailing lists are required periodically. For example, to set up a script called 'mailpr' the following command can be stored in a file created under ed.

```
sort <$1 |lpr
```

The script file is granted execution permission with the chmod or 'change mode' command, and is thereafter processed by the shell, utilities and kernel as though it were a utility or system file itself.

6.9.2. Positional Parameters.

This example uses 'positional parameters' which allow a script to act in a generalised fashion. They are used to specify the position of more specific data to be entered later. These actual data are entered in the form of arguments to the command that executes the script.

Typing

```
% mailpr addlist1
```

will cause the shell to execute the mailpr script, using the addresses contained in addlist1, by substitting addlist1 for $1. If the script had contained $2 or $3 the shell would have substituted any second or third word given to the command. Upto nine positional parameters ($1 to $9) may be used in this way.

6.9.3. Control Flow.

Like high level languages such as Pascal and FORTRAN, the shell programming language includes logical structures for control flow. The structures available are:

if...else
while
foreach or for
case or switch

For example a script called 'tel' containing:

```
for i
do grep $i /usr/lib/telnos; done
```

can be activated by the command:

```
% tel hello
```

to list all the lines containing instances of the word 'hello'.

The existence of these constructions, and the editable nature of existing command files, means that new command systems can be developed by piecing together and amending existing shell programs.

6.10. Process Control.

Any operation currently in action is a "process". The normal sequence of processes follows the order of command input, whether from an input device or a shell script, but the MG-1's process control systems allow more complex arrangements.

Background processes (commands ending in '&') execute while other processes are directly controlled by the user. While freeing system resources for other activities, background processes have the disadvantage of reducing user control, especially when they have to be terminated prematurely.

To halt a normal process, the <Delete> key or the <Ctrl> <\> sequence is used. However, some processes such as those operating in "raw" mode (such as screen editors) cannot be interrupted in this way. Instead, they require the use of the 'kill' signal which terminates processes 'with extreme prejudice'. Successive depressions of the <Ctrl> <Alt> <Esc> combination sends progressively stronger kill signals: SIGINT, SIGQUIT, and SIGKILL. In most situations, the SIGINT signal will be sufficient to halt a process. For the handling of runaway processes, see section 11.5.

Using SIGKILL frequently resets the system back to the login prompt, and should be used with caution, as all current processes will be halted.

6.11. Graphics Facilities.

GENIX incorporates tools specifically designed for the high resolution graphics capability of the MG-1. The window manager, for example, provides a set of commands for displaying multiple windows. The screen can be divided into several, perhaps overlapping, areas. Commands are available to change the size and position of windows and their order of overlap. Areas within windows can be 'sensitised' to receive input commands from a pointing device such as the mouse.

GENIX also includes a range of graphics library routines designed to simplify the task of producing application packages. Because the kernel is the basis of graphics support on the MG-1, and because the operating system allows the addition of new commands and utilities, the graphics systems can be radically updated.

6.12. The On-line Documentation and Learning Aids.

GENIX offers a number of on-line aids in addition to the printed Programmer's Manual. The level of provision is determined by the hard disk configuration because the complete manual requires 1.4 Mbytes of disk space.

The on-line version of the GENIX Programmer's Manual is accessed with the 'man' command. A number of arguments may be added, for example a subject heading:

```
% man date
```

produces a copy of the manual section on 'date'. The full range of options avilable to this command is given in section 1 of Volume I of the GENIX Programmer's Manual, or may be displayed by typing:

```
% man man
```

A further facility is 'apropos' which lists the manual sections containing instances of any of the keywords in the command title. Embedded instances of keywords are detected. For example references to "compiler" will be listed searching for "compile". The case of characters is ignored.

Chapter 7
A GENIX Session

7.1. Introduction.

This chapter is intended to introduce some of the most commonly used GENIX commands. It draws on the information laid out in Chapter 6, and relates it to the development of a typical application, in this case, a series of text documents and programs.

The author of this document intends to create a number of files, each containing a separate chapter of the document. This set of files will be held under its own directory entry. In addition, a series of memos to colleagues, an appointments diary, and a series of example programs will be created. Each of these will have its own directory entry, and may contain a number of files.

7.2. Login.

After powering up, the MG-1 displays the login message and prompt. The user types in a user name, hereafter given as "john", followed by a carriage return.

It is at this point that the first of the MG-1's security systems may be encountered. When appropriate, the password prompt will appear. Full details of the password system are given in Chapter 10. If this prompt should appear, the user types in the password, and presses <RETURN>.

In either case, the last login date is displayed, followed by the shell prompt. The default configuration is the C Shell, which uses the **%** symbol as its prompt. At this point, the user may decide to change to the Bourne Shell, which uses the **$** symbol. Details of the chsh command are given in section 6.4, but this example assumes that the C Shell is in use throughout.

From the login point onwards, only a portion of the screen is active. The Window Manager documentation that accompanies this Guide gives full details of the routines needed to activate the window systems available. As windows are created and manoeuvred around the screen, more of the screen area will be used.

User commands are executed from the command line buffer only after the <RETURN> key is pressed.

7.3. A GENIX Session.

The following diagram represents the directory structure that will be created, including the home directory.

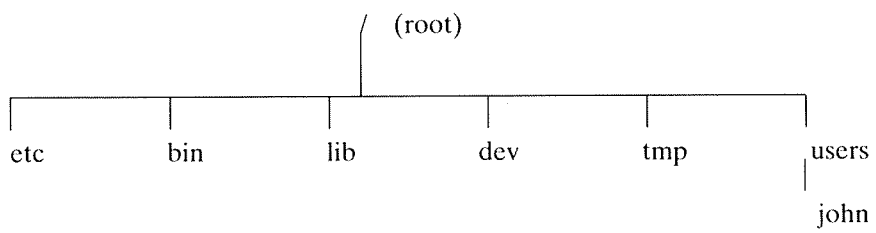


Figure 7.1. The User's Home Directory.

For a description of the system directories, refer to Chapter 6, Figure 6.2.

John intends to create and work on the file/directory structure illustrated in Figure 6.2.

It should be remembered that all commands are terminated with the <RETURN> key. The <RETURN> key is located on the right hand side of the alphanumeric section of the MG-1

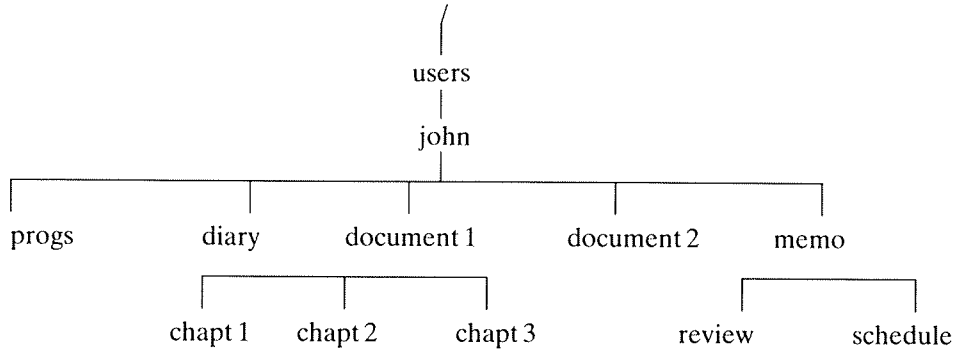


Figure 7.2. The User's Directory.

keyboard, labelled with a clockwise arrow.

The appropriate use of the <RETURN> Key is hereafter assumed, and not explicitly mentioned when describing the use of shell or utility commands.

It is inevitable that typing errors will occur. Character deletion is achieved by pressing the back-space key (located above the <RETURN> key and labelled with a left arrow); word deletion is achieved by holding down the <Ctrl><w> keys; deletion of a whole command line is obtained using <Ctrl><x>.

Filenames can include as many as fourteen characters. It is advisable to restrict the choice to upper and lower case alphanumeric characters and avoid all punctuation marks except the full stop (.).

GENIX distinguishes between upper and lower case characters within filenames. Accordingly, 'Chapt1' and 'chapt1' are not the same file.

Files with different pathnames are different files. This means that '/users/john/file1' is not the same file as '/users/mary/file1' even though, from their respective home directories both users could simply refer to a file called "file1".

The full pathname of a file is required if the file to be accessed is not held in, or about to be created in, the user's current working directory. However, it is usually more convenient to be in the directory to be worked on, than to work from some other directory.

Many of the commands available from the shells have a range of possible arguments. In many cases, these complement each other, and are therefore to be combined. The full syntax of such commands is listed under the appropriate sections of the GENIX Programmer's Manual, but a common form is as follows:

```
% cat –nb
```

For an explanation of this particular command, see section 7.3.5, below.

When the arguments are filenames, each entry should be separated by a space.

However, when a command uses the same filename as consequtive arguments, the '.' abbreviation may be used. The command

```
% cp /flp/myprog.c .
```

has the effect of duplicating the last-used filename and producing the same effect as

```
% cp /flp/myprog.c myprog.c
```

See section 8.18 for an explanation of this particular command.

Note that all commands that involve files and directories, for example deletion, reading, copying, appending, and moving, are subject to the user having appropriate access permission. See Chapter 9 for further details.

Commonly used sequences of commands can be produced automatically by reprogramming MG-1 keys. This procedure is described in section 5.8.4.

7.3.1. Determining directory position: pwd

A user's position within the file structure can be determined with the 'print working directory' command typed against the shell prompt :

```
% pwd
```

The system response to this command will display the pathname of the current directory, for example:

```
/users/john/memos
%     .
```

This indicates that John's current working directory is called 'memos'. Note that the shell has completed the operation by returning to the prompt.

7.3.2. Creating a directory: mkdir

The mkdir command creates a directory (but not files) made up of the names used as arguments to the command. It should be remembered that a directory is simply a home for files or other directories, and is not a program or block of text, for example.

These entries are appended to the bottom of the branch currently 'occupied' by the user. For example, if the user is situated in the home directory, a command taking the form:

```
% mkdir progs diary document1 document2 memos
```

will create these directory entries, as shown on level 4 of Figure 7.2.

7.3.3. Creating a file: cat >

The 'cat' command is an abbreviation of "catenate", and stores any typed input in a named file, here called 'monday'. If the file does not already exist, it is created by the cat routine. If it does already exist, existing text is overwritten. Accordingly, care should be taken that useful text is not lost. When text is being entered under cat >, the shell prompt is suspended. In order to signal the end of the text, and to return to shell control, type in the <Ctrl><d> sequence which generates the code for 'End of File'. The system responds by displaying the shell prompt.

To create a memo or diary entry, John should move to the appropriate directory with the 'cd' command, and type the following:

```
% cat > schedule
Remember: dentist at 15.30 today
<Ctrl><d>
```

7.3.4. Appending to a file: cat >>

To append to the end of an existing file, for example, 'schedule' use the cat >> command:

```
% cat >> schedule
Take the rest of the day off
<Ctrl><d>
```

The shell prompt is again suspended until data input is complete, and <Ctrl><d> is entered.

Combinations of files can be created by catenating files with these commands:

```
% cat file1 file2 >> file3
```

has the effect of catenating two files and appending them in sequence to the end of file3.

7.3.5. Viewing the contents of a file: cat

The cat command as used without any directional indicators ('>' or '>>') displays the contents of a named file. This command operates by not supplying a specific destination; the shell supplies the default logical destination, which is the screen.

A number of arguments to the command, in addition to file names, are available. For example:

```
% cat –n schedule
```

causes the output lines of the file 'monday' to be numbered sequentially from 1. The –b argument causes numbering to exclude blank lines. A full list of cat command options is available in the GENIX Programmer's Manual.

In the example, the following sequence would occur:

```
% cat schedule
Remember: dentist at 15.30 today
Take the rest of the day off
%
```

7.3.6. Viewing the contents of a file: more

Another way of examining the contents of a file is the more command:

```
% more review
```

By typing this command, the file is displayed one screen page at a time. At the bottom of each screenful of text, the legend "**---More---**", followed by a field that displays a percentage is displayed. The percentage field shows how much of the total file has been displayed. Scrolling is suspended, and the next page is viewed by pressing the <space bar>.

7.3.7. Changing directory position: cd

Users can alter the current working directory by means of the change directory command, cd, plus a directory pathname as argument.

Issued without arguments, the cd command returns the user to the home directory.

Two abbreviations are available for use within pathname declarations. One dot '.' always refers to the current directory; two dots '..' always refers to the directory immediately above the current directory. These abbreviations are integrated into pathnames in exactly the same way as normal file or directory names.

The following command is used in order to move from the current working directory (/users/john/memos) to the document1 directory (/users/john/document1):

```
% cd /users/john/document1
```

or

```
% cd ../john/document1
```

Note the use of the abbreviation feature. In this particular example, it would be permissable to simplify the command to:

```
% cd document1
```

because the new position is only one level below the current position.

7.3.8. Using the line editor: ed

This line editor is a complex and comprehensive text handling system which includes a full range of searching, substitution, deletion, insertion, block alteration, metacharacter and "wild card" facilities. Accordingly, the following is a brief survey only, and is not intended to be exhaustive. Full details are given in the commands section of Volume I of the GENIX Programmer's Manual, and in the Text handling section of Volume II.

To invoke the line-oriented editor, type:

```
% ed
```

While ed is active, the shell prompt is suspended, and although nothing appears to be happening on the screen, the editor is waiting for a command.

The commands required to create a simple text file are as follows:

```
% ed              *To invoke the editor
a                 *The "append data" command
Remember: dentist at 15.30 today*The input text
.                 *The "end of text" marker
w chapt1          *The "write filename" command
33                *Characters in the input data
q                 *The "quit ed" command
%                 *The shell prompt
```

The number of characters in the input data is a variable returned by ed after writing the text to the specified file, and includes spaces and carriage return characters.

The append command is also used to input additional data to an existing file. Where an existing file is concerned, the filename specifier is displaced to the beginning of the command sequence:

```
% ed chapt1
a
Take the rest of the day off
.
w
30
q
%
```

Although the shell prompt is disabled while editing a text file within ed, shell commands may be used by preceding each command by the '!' (shriek) character. For example,

```
!pwd
```

would return the current work directory location. Although this information is displayed on the screen, it is not included when the input text is written to the specified file by the editor.

7.3.9. Directory listings: ls

The ls command lists the titles of files contained in a directory in alphabetical order, for example:

```
% ls /users/john/memos
review
schedule
%
```

Full details are given in the GENIX Programmer's Manual. However, certain of the arguments to ls are so frequently used that they should be mentioned here.

The –a argument lists all the files, including those prefixed with '.' and '..', which are normally omitted from the listing. The –l option provides a 'long' listing which includes file length in bytes, the owner, and the time of last update. –t sorts by time of last update, and –r reverses the order of any sorting procedure.

To ascertain the existence of a particular file, supply the filename as an argument. The system reports either the filename, if found, or the message "(filename) not found".

7.3.10. Moving a file between directories: mv

The mv command moves a file from the current working directory to some other directory specified as an argument to the command. This procedure removes the file from the current directory. The command:

```
% mv chapt2 /users/john/document2
```

moves the file chapt2 from the current directory to the document2 directory.

7.3.11. Renaming a file: mv

The mv command is also used to rename a file. In this event, the two filename arguments to the command specify files within the same directory.

```
% mv schedule newschedule
```

renames 'schedule' as 'newschedule'.

It is important to ensure that new filename is not the name of an existing file. An existing file with the same pathname will be overwritten.

7.3.12. Copying a file: cp

Copying a file to another directory with the cp command leaves the file in its original directory as well as creating a copy. The command

```
/users/john/document1/chapt3 /users/john/document2
```

typed against the system prompt creates a copy of chapt3 in the document2 directory. This operation can be carried out irrespective of the user's current work directory.

```
% cp chapt3 document2
```

performs the same task when the user's current directory is document1.

Remember to update both copies of a file: making changes to one file does not automatically change the other.

7.3.13. Removing a file: rm

The rm command removes or deletes files whose names are supplied as arguments, from the file structure.

It is advisable to use the rm command with caution in order to avoid the accidental loss of valuable information. Check the file contents first.

```
% rm newschedule
```

deletes the file from the current directory. To delete files held in directories other than the current work directory, specify the pathname of the file. For example, from the document1 directory, type:

```
% rm john/memos/newschedule
```

A full range of options is available; for details, see section 1 of Volume I of the GENIX Programmer's Manual.

7.3.14. Removing a directory: rmdir

Removing a directory involves two steps: deleting all the files in the directory and then deleting the directory itself.

Prior to removing a directory, use the ls command to list the files to be deleted. Make sure that the files to be deleted do not contain information that should be retained.

To remove all the files in the current working directory type:

```
% rm *
```

Use the cd command to move to some other directory, for example:

```
% cd ..
```

which has the effect of moving up the tree structure by one level, and type:

```
% rmdir
```

followed by the directory name. For example:

```
% cd john/document2
% rm *
% cd ..
% rmdir document2
```

When attempting to use the rmdir command to remove a directory, the system may reply that the directory still contains files even though the rm * command has been used and ls does not indicate any further files. This message usually means that there are files beginning with '.' such as .profile, left in the direçtory.

The ls command in its simple form does not list '.' and '..' files, and they cannot be accessed by the wild card (*) metacharacter. To obtain a listing of all 'dot-prefixed' system files use 'ls –a'. Standard and system files will be listed.

System files prefixed by '..' may not be deleted so their removal is not neccessary in order to remove a directory. To delete all the '.' files use:

```
% rm .*
```

Remember that the deletion of a directory and its component files depends on the user having the appropriate access permissions. See Chapter 9 for details.

7.3.15. Creating a Shell Script.

A script is an executable file containing a sequence of instructions. In this example, the user requires a procedure that will feed the contents of a series of text files from a specified directory into the GENIX spelling checker. The output from the spelling checker (unrecognised words) must be piped into the 'pr' print layout facility to create a three column format which is in turn directed to a series of new files within the existing directory. It is envisaged that these processed files will be used for proof-reading, perhaps by analysing the most commonly misspelt words, and running a global 'search and replace' operation.

This routine can be held in a single executable file, which, in this example, will be called "spellcheck", stored in a directory called "scripts". Once completed, this file may be executed as a background job to automatically check and format a text file while another is being edited by the writer.

The sequence required to establish such a shell script is as follows:

```
% cd                                                  *To return to the home directory
% mkdir scripts                                      *To create the directory
% cd scripts                                         *To enter the new directory
% ed                                                 *To access the line editor
a                                                    *The "append" command
cd $1; spell –b chapt$2 | pr –t3 > checkchapt$2      *The script command sequence
.                                                    *The "end of text" marker
w spellcheck                                         *The "write file" command
48                                                   *Character count system response
q                                                    *To quit the line editor
% chmod x spellcheck                                *To assign access permissions to
                                                     the script file and to make the file
                                                     executable. For further details,
                                                     see section 10.8
```

For a full list of the options available to the commands used here, see the GENIX Programmer's Manual.

7.3.15.1. The Use of Positional Parameters.

In the above command sequence, a shell script has been created that will act in a generalised fashion upon any file beginning with 'chapt', held in any directory, and will modify its contents to produce a new file whose name begins with the string 'checkchapt'.

The first and second parameters respectively specify the data to be entered into $1 and $2. Duplication of such items is permissible, as the example shows: the double use of parameter $2 ensures that the files containing the initial and the processed text have the same identifiers.

7.3.15.2. Executing a Shell Script.

Shell scripts may be invoked in a number of ways, for example by specifying the script as an input file to the shell by means of:

```
% sh spellcheck
```

or:

```
% sh < spellcheck
```

In the third case the user must have execute permission for the file. Execute permission is set by the chmod command as used in the example:

```
% chmod x shellscript
```

Once the execute permission has been assigned, the command to invoke the script is simply:

```
% shellscript
```

Where the script contains positional parameters, their contents should be specified as arguments to the invoking command:

```
% shellscript document1 3
```

On completion, spellcheck will have created a new file, called 'checkchaptn' (where 'n' is the chapter number) in the specified directory. In the example given above, the script inputs document1/chapt3 and produces document1/checkchapt3. The new file will contain a 3 column list of all the words in the specified chapter which are unrecognised.

In order to execute this script as a background job while perhaps editing a further text file, type:

```
% shellscript document1 3&
```

The shell prompt is repeated and the editor, for example, may be used as normal.

7.3.16. Compiling and running programs.

The MG-1's operating environment is capable of supporting a wide range of programming languages such as C, FORTRAN and Pascal. Software systems may comprise interlocked modules of different languages.

C is a general purpose programming language, and is the source language for much of GENIX itself. Because it is technically a relatively low level language, C deals with the same level of data objects as the MG-1 itself, that is characters, numbers, and addresses. C itself supplies no I/O handling facilities such as READ or WRITE statements, or composite data object handling routines for strings, sets, lists or arrays. Many of these facilities are provided by the standardised function library supplied with the compiler.

However, the efficiency of C in low level data handling means that there is little need for the addition of Assembler routines to C systems.

The C compiler is central to the GENIX operating system, and is invoked by the command:

```
% cc
```

which has many options. By default, the executable file produced by cc is called a.out.

To compile and run a C program (called for example ex1.c) held in the current working directory (for example john/progs), the appropriate commands are:

```
% cc ex1.c
% a.out
```

The option –o allows the user to specify a particular filename rather than relying on the "a.out" default. The command takes the form:

```
% cc –o assignedname filename
```

For example, to compile and run a program to be called 'example1', type the commands:

```
% cc –o example1 exc1.c
% example1
```

A program can be composed of several files which are compiled together by specifying each filename as an argument to the –o option:

```
% cc –o newprog ex1.c ex2.c ex3.c
ex1.c:
ex2.c:
ex3.c:
%
```

and executed by typing its name:

```
% newprog
```

The program that is executed consists of the three sub-programs that were compiled together.

A full range of documentation accompanies each of the optional MG-1 compilers, such as Pascal and FORTRAN 77.

7.3.17. Sending messages: mail

The 'mail' facility allows any user to send messages to any other user whose user-name is known. For example, to send a message to user "sue", type:

```
% mail sue
Congratulations on excellent job
```

During editing, the shell prompt is suspended until the <Ctrl><d> sequence is used to signal 'End of File'.

Typing the mail command without an argument accesses any messages that may have been sent to your account by other users.

7.3.18. Log Out.

To log off from the system, type:

```
% logout
```

To switch off the system, type:

```
% off
```

In order to combine the two commands, separate them with a semi-colon:

```
% logout;off
```

Chapter 8
Interactive Graphics

8.1. Introduction.

8.1.1. Uses of Interactive Graphics.

Two broad groups of computer graphics systems are distinguishable: systems used for information display such as bar charts, graphs and text processing; and systems used to illustrate artwork, modelling or physical objects and processes. Computer-aided design (CAD), engineering (CAE), and manufacture (CAM) all rely heavily on graphics systems. Integrated circuit design, molecular modelling and flight simulation are other major uses. The techniques used to create such applications are described in section 8.3.

The MG-1 32-bit supermini graphics system provides the high-resolution graphics capabilities needed to handle such applications. The Virtual Memory based raster scan system provides a full 16 Mbytes of storage for each graphics program, and does away with the need for the high cost vector generation techniques used by the more traditional systems. The graphics library uses the raster system to handle graphics objects while in turn acting as the basis of high level graphics packages. The Window Manager allows the control of a number of concurrently active virtual terminals.

8.1.2. Raster Scan v Random Scan.

Both styles of graphics system involve building an image from the basic picture component, the "pixel". Each pixel must be assigned an intensity value; those set to '1' are switched on, and appear black on the screen: those set to '0' are off and remain a part of the background. The MG-1's raster scan system of graphics handling stores pixel intensity values in a two-dimensional array in memory. These arrays of information in memory are called "rasters".

The pixels to be switched on or left off are identified by the functions provided in the MG-1's graphics library. Each time a pixel is processed, the pixel setting function is called to assign the appropriate intensity value. The library provides the means for creating rasters containing any picture element that might be required. By creating, manipulating and combining the rasters created in this way, whole pictures can be built up. The final picture may contain any mixture of points, lines, polygons, solid areas and text characters.

In contrast, the random scan system uses a set of graphics drawing functions to calculate the intensity value of each pixel, and stores them in a sequential file. They are then displayed in that order, one at a time.

8.1.3. The MG-1's Graphics Capabilities.

8.1.3.1. The Screen.

The display system consists of the memory required to contain the image in raster form, a variety of logic devices, and the hardware required to generate the image. Electrons generated by a cathode ray tube are fired onto the inner surface of the screen. Focusing coils determine the precise point of the beam on the screen, and the intensity of the beam determines whether the pixel is on or off. The intensity values assigned by the drawing functions are used to control the beam intensity.

Because the light produced by the phosphor fades at a known rate, the image must be refreshed in a regular cycle. The MG-1's refresh cycle is repeated 57 times each second. This is fast enough to prevent flicker, and to allow the phosphor to display considerable detail without image smudging.

8.1.3.2. The Cursor.

An interactive graphics system involves an image produced as output, and a variety of user responses channelled into a program by way of the mouse or keyboard. Both input systems require some form of cursor to track input events. The MG-1 stores a number of cursor rasters in a separate part of the memory. They may take a variety of forms, but are all contained in a 64×64 pixel raster. The Panellist (for more details see section 8.5.1.1) ensures that this size requirement is fulfilled by truncating or white-padding. The MG-1's cursor system is hardware based as it uses an I/O processor (see section 3.2.1.6) to mix in the appropriate cursor image by way of XOR or OR logic.

8.1.3.3. Rasterops.

A raster has been introduced as a two-dimensional array of pixels that completely defines a component part of a picture. For example, a screenful of playing cards may be defined by a series of raster representations such that the bare body of a card is stored in one raster, and a diamond-shaped pip in another. These two rasters can be combined to produce the image of a card with between one and nine pips (or more if necessary). The digits would be stored as a text font held as a separate file of rasters. A full set of logical operations are available that will produce any required image consisting of a combination of these rasters. Because each raster has an identifier, a graphics program can easily call up the appropriate combination.

A raster system requires two components, a representation system for the naming and creation of rasters, and an operation system for their modification and combination. Raster creation is handled by the MG-1's graphics library which specifies routines for the creation of lines, arcs, circles, points and text characters. Each of these is described in section 8.2.

Raster combination and handling is controlled by way of the MG-1's set of sixteen boolean operations between two rasters. In general, two rasters, called "source" and "destination" are involved in a rasterop. Both are used as inputs to the function, and the output is placed in destination.

A rasterop is defined as

RasterOp(Op, Source, SPos, SSize, Dest, DPos, DSize)

where Op is a value between 0 and 15, selecting one of the sixteen logical operations that may be performed between the source and destination rasters. The other six arguments define the identity , size and position of the two rasters, here called Source and Dest. The sixteen operations are as follows:

0	F_0	clears to 0 (white)
1	F_NOR	not (source or dest)
2	F_NDS	(not dest) and source
3	F_NOTD	not dest
4	F_DNS	dest and (not source)
5	F_NOTS	not source
6	F_XOR	source xor dest
7	F_NAND	not (source and dest)
8	F_AND	source and dest
9	F_NXOR	not (source xor dest)
10	F_S	source
11	F_NDORS	(not dest) or source
12	F_D	dest
13	F_NSORD	(not dest) or source
14	F_OR	source or dest
15	F_1	clears to 1 (black)

There is a further raster handling operation available. The BatchRasterOp function uses a list of source rasters to perform a series of rasterops between each source and the named destination. The source rasters are listed in the null-terminated argument SList. The batch function takes the form:

BatchRasterOp (Op, SList, Dest, DPos, DSize)

Op is again one of the sixteen logical operations listed above.

A common need is the dynamic creation of rasters. This may be achieved using the following:

NewRaster (w, h)

where the two arguments specify the height and width of the raster. When the creation process is complete, all pixels will be set to white.

Sub-images of existing rasters can be created using

SubRaster (Parent, x, y, w, h)

where the Parent argument identifies the existing raster, and the other arguments specify the origin (in parent coordinates), and size of the sub-raster.

Rasters created with these processes can be destroyed using

FreeRaster (Raster)

This function should not be used on a sub-raster whose origin coincides with the origin of the parent.

8.1.4. Methods and Techniques.

8.1.4.1. Plotting and Coordinate Systems.

Points are displayed on the screen according to the physically addressable locations available. Because the MG-1's screen measures 1024×800 pixels, there are 819,200 discrete locations. This is the "resolution" of the screen.

The resolution of the screen is a physical parameter of the system. The "precision" of the system is a logical parameter in that it determines how well the resolution is used. Precision is determined by the number of physical pixels that can be actually accessed. If memory is too small to maintain an adequate system precision, many of the screen locations will not be accessable, and so a high-level resolution would be wasted. On the other hand, if the precision is much greater than the resolution, there will be addresses held in memory that do not correspond to available pixels, and so much of the potential image detail will be wasted. For this reason, precision and resolution are equal.

The basis of a point plotting system is the cartesian coordinate system that uniquely identifies each discrete point on the screen. The MG-1's display system ranges from 0 to 1023 on the x axis and 0 to 799 on the y axis, and provides the space onto which application coordinates are mapped. This is introduced in the next section, and covered generally throughout this chapter.

8.1.4.2. Transformations.

Transformations are the general class of image-handling techniques that produce picture changes in an orderly and controlled fashion. They depend on standard and well-understood mathematical techniques such as trigonometry and matrix-handling. The commonest transformations are mapping between one coordinate system and another; rotation; translation; and scaling. All transformations may be concatenated to produce multiple operations to be performed with a single function.

One of the great advantages of transformations is the opportunity to use different coordinate systems. This allows a number of non-standard functions to be applied to a picture within the context of an appropriate coordinate system. The final product of these transformations is then mapped into screen space only when the appropriate visibility guarentees have been applied. A general discussion of coordinate mapping systems follows throughout this chapter; the section on clipping is especially important.

Geometrical transformations allow an image to be manipulated by altering the position and scale of its component parts.

A detailed discussion of the various geometrical transformations appears in section 8.2.9.

8.2. Basic Methods.

8.2.1. Introduction.

Graphics handling systems are based on the pixel as the smallest unit of image production. To create a picture, many hundreds or thousands of pixels must be set to an appropriate intensity value. Lines, arcs, characters and solid areas are all made up of single points.

Composite entities of this sort are created using the MG-1's Graphics Library. These functions are covered in the following sections. In all cases, the function relies on a pixel setting function to actually translate the calculated image into physical pixel intensity settings.

8.2.2. Point Plotting.

Situations requiring the modification of a single pixel within a raster may make use of the point plotting call which takes the following form:

GPlot (Op, Brush, Paper)

where Op defines the logical operation performed, Brush defines the location, and Paper identifies the relevent raster.

The inverse of this function is the pixel interrogation function

GPoint (Brush, Paper)

which returns the intensity value of the pixel at Brush within the raster called Paper, giving 0 if the pixel is set to white, and 1 if set to black.

8.2.3. Line Drawing

Line drawing techniques are essential to any computer system, and are used in block diagrams, bar charts and graphs, engineering and plans, and any number of other applications. Curves may be produced as an approximation of short straight line segments.

Lines are created from an aggregate of points. Points are calculated from the parameters supplied to the line drawing function

GLine (LineEnd, Op, Brush, Paper)

where Brush is the current drawing position, and LineEnd specifies the end point of the line to be produced. A line is drawn between these two points in the raster called Paper. LineEnd then becomes Brush, irrespective of any clipping that may take place. A series of techniques are used to minimise the stepping effect produced when an idealised diagonal line is translated into a finite number of discrete points on a screen. The pixel setting routine assigns an appropriate intensity value to the pixels involved, 0 for the background and 1 for the line.

The success of a line drawing system is assessed in terms of the straightness of the lines produced, the accuracy with which lines meet, whether or not the lines have a constant density, irrespective of angle and length, and the speed at which drawing occurs.

8.2.4. Moving the Brush.

To move the position of Brush without drawing a line, use the function

GMove (LineEnd, Brush)

where LineEnd is the current position, and Brush is the desired point.

8.2.5. Curve Generation.

Arcs may be placed in a raster using the GArc function which takes the form

GArc (ArcCentre, ArcEnd, Op, Brush, Paper)

GArc operates in the same way as the GLine function, drawing from Brush to the boundary of a rectangle whose corner is identified by ArcEnd. Drawing procedes in a clockwise direction around the point ArcCentre. If no boundaries are encountered, the function produces a complete circle, which is then clipped to the raster Paper.

Full-circle generation is handled by the GCircle function:

GCircle (Op, Radius, Brush, Paper, Clip)

A circle of radius Radius is drawn around the current Brush position, and clipped to Paper if Clip is non-zero. If Clip is not set, the drawing speed increases, but at the cost of possible segmentation violation and overwriting of the user's address space when the circle is drawn outside Paper.

8.2.6. Solid Area Filling.

Solid areas can be created by establishing a series of lines within a raster using the above functions. These lines are used to set up the boundaries of the polygon. In order to in-fill this area, the GAreaFill function is available:

GAreaFill (Brush, Paper)

The function floods the polgon outwards from the brush position until it reaches the boundaries of the polygon. The boundaries are identified when pixels are reached that are the inverse of the tone at the starting point. The polygon is then flooded with this inverse tone.

8.2.7. Character Generation.

The raster system's ability to handle arbitrary patterns of pixels makes it ideal for the displaying of characters.

The MG-1 provides a series of pre-defined fonts. These are contained in the directory /usr/lib/mfont. Also provided is a font editor which allows the user to alter the appearence of any character by way of the mouse. For access to the font editor, type the 'fonted' command followed by the name of the font to be edited, for example:

```
% fonted elite
```

A menu of the characters available within the selected font is displayed. The mouse is used to manoeuvre the cursor over a particular character, and a mouse button clicked. The selected character is then displayed as a raster-style diagram made up of a matrix. The character is picked out in black squares, the background in white. To alter the appearence of the character, use the mouse to manoeuvre the cursor onto a square and click a button. Black squares become white, and vice versa. To save this new design, type

```
% save
```

To return to the standard design, type

```
% undo
```

To create a new font, make a copy of an existing font, and make changes to the copy using the font editor as described above.

The main advantage of a raster-based character system is that the handling of descenders and proportional spacing is an easy matter. These are much more complex problems on a random scan system. Each character is stored in a raster of fixed height, but variable width. This arrangement compensates for the letter "i" being much thinner than the "w", and ensures that spacing is equal, irrespective of this difference.

By allocating a raster coordinate system with positive and negative y axis values, the raster's origin may be aligned with a display base-line such that characters with descenders are set below those without. This is a primary example of a convenient coordinate system being used to create an image before it is mapped onto screen space for display.

The MG-1's graphics system includes a number of functions for displaying characters under program control.

GPutC (c, Op, Brush, Font, Paper)

writes the character c from Font onto the raster called Paper. Brush identifies the left-hand end of the character's baseline. Op specifies the boolean operation that will be used to combine the character with the destination raster. After the character has been drawn, Brush will be updated to

(x+w, y) where w is the width of the character.

The GPutString function draws strings more efficiently than repeated calls to GPutC, and takes the form

```
GPutString (Paper, Brush, Op, Font, String)
```

String is written to Paper at Brush according to Op, using characters from Font.

Fonts are read by either the GFontRead call or the ReadRasterFontFile call. Fonts are held in vfont format files. Each file contains a variety of header information, an array of 256 character descriptions, and their associated bit maps. Fonts are loaded into a format more suited to rapid output; two formats are available.

The GFontRead function is the most commonly used, and is used with fonts held in rfont format. Characters are defined in this form by way of a baseline, centre line, width, and height. The ReadRasterFontFile call is based on ofont formatted files, and is used to access a font held in the form of a single raster. Individual characters are then addressed as sub-rasters with their own offset address within the font raster.

By selecting characters from a font set, using one of the above methods, and mixing them with other raster pictures, annotated diagrams can be produced.

Full details of the character handling facilities can be found in the graphics(1), fprint(3), ofont(5), rfont(5) and vfont(5) references in the GENIX Programmer's Manual.

8.2.8. Scan Conversion.

Scan conversion is the process that translates a line or point or text character from a simple geometrical expression into a raster representation. The drawing functions calculate the positions of the pixels to be set on or off, and call the SetPixel function which takes the following form:

```
SetPixel (Raster, x, y, Intensity)
```

where the first argument identifies the raster, and the next two identify the (x,y) coordinates of the appropriate pixel.

The scan conversion system called by a library function is very efficient for the creation of long lines or other pictures. However, where short lines are required, it may be more efficient to use the SetPixel function directly.

Solid areas are an integral part of many graphics applications, such as technical illustration and animation. Three properties of a solid area lie at the heart of its creation: its mask, its shading, and its priority.

The mask of an object is the pattern of pixels set to on and off to create the image in raster form. The process of determining the mask is called "solid area scan conversion".

The shading of an object specifies how the intensity of each pixel is asssigned.

The priority of an object resolves the problem of overlapping objects; lower priority objects are obscured by higher priority. It will be apparent that this factor is not important when overlapping does not occur, or where overlapping causes pixels of the same intensity value to correspond. However, when white and black pixels coincide, the situation is resolved according to the priority of the overall object within the image.

A simple algorithm for scan converting solid objects might take the form:

```
WriteRectangle (x, y, width, height, intensity)
```

Such algorithms relt on the fact that discrete pixels obviously have integral coordinate values: the call

```
WriteRectangle (8, 5, 3, 4, intensity)
```

will scan the rectangle 8 <=x <=11,5 <=y <=9 with the given intensity. This particular algorithm is unrealistically simple, but gives an indication of the processes involved.

More complex shapes use complicated polygon scan conversion algorithms. While a large number of these are available, a typical method is to define a polygon in terms of its vetices. Pixels occuring on an imaginary line scanning the polygon that has accrued an odd number of intersctions, are

evidently within the polygon. These pixels are valued accordingly. Those on a line currently with an even number of intersections are obviously outside the polygon, and are assigned the background intensity value. A number of refinements to this general pattern are used to improve performance.

8.2.9. Transformations.

Many graphics applications involve part or all of the picture undergoing a change in size and orientation. Such transformations are accomplished by way of standard mathematical techniques such as coordinate geometry, trigonometry and matrix-handling. These transformations are then combined with the scan conversion routine to compute the appropriate pixel values after, for example, a scaling and a tranlation.

A transformation is a single mathematical entity, and may be represented and handled using a single identifier. However, transformations may be concatenated in order to produce a more complex result. It is essential to realise the importance of maintaining the order of transformations if they are to be combined. The effect of a rotation through 90° followed by a translation in space by Tx=-80, Ty=0 is very different to the result of the two in reversed order.

The basic transformations available are mapping from one coordinate system to another; rotation; translation; and scaling. The first is an integral part of Window Manager and high-level operations. Such functions as PanelUpdate are described in section 8.5.

The general form of a translation is: x'=x+Tx y'=y+Ty

where (x',y') are the amended coordinates, and T is the displacement along the relevent axis.

Rotations take the form: x'=x cos θ + y sin θ y'=-x sin θ + y cos θ

where θ is clockwise angle of rotation.

Scaling transformations take the form: x'=xSx y'=ySy

where S is the scaling factor. For example, if Sx=Sy=2, the image is doubled in size. If Sx and Sy are not equal, the image will be distorted. Negative Sx and Sy values produce mirror images.

8.2.10. Clipping.

Many graphics applications, especially those employing the Window Manager, require the image to be viewed a portion at a time. This effect is noticable, for example, when the image undergoes an upward scaling such that part of the picture grows beyond the display area's boundaries. The process that retains the visible area and 'discards' the invisible, is called clipping.

A clipping algorithm may be used to analyse an image for its visible portion. This visible portion is then mapped onto the screen space coordinates. In very general terms, a clipping algorithm uses a pair of inequalities to ascertain whether a point (x,y) is visible:

xleft <=x <=xright ybottom <=y <=ytop

8.3. Basic Interactive Techniques.

8.3.1. Introduction.

The designer of an interactive application makes extensive use of such interactive techniques as pop-up menus, rubber-band lines, methods of object selection, and dynamic rotation. These techniques are the 'raw materials' of user interface design. They have been developed over the years, and some of them date back to the earliest days of R & D activity. For evidence of the wide variety of techniques available, see the graphics references in Appendix F.

The applications programmer, working with a high-performance work station like the MG-1, expects to be able to choose freely among the repertoire of commonly used techniques. However there are certain techniques that make heavy demands on workstation resources, for example "dragging" large areas of the screen image around with the mouse, dynamically adjusting the shape of spline curves, or running complex animation programs. Some techniques are difficult to support under a standard UNIX environment; this is the case for example, with most window-related operations.

It is important to ensure that the workstation can support a reasonably wide range of interactive techniques. Unfortunately this cannot be done through any single measure, such as increasing the

speed of rasterops; this is because each technique has its own implications for resource requirements.

The only way to ensure that the workstation can support a wide range of facilities is to consider the full range of these requirements.

The purpose of this section is to define the range of interactive techniques likely to be found in MG-1 applications. It is essentially a catalogue containing a brief description of each technique, sufficient to allow analysis of the full scope of resource requirements. It is not intended to assist the user interface designer and it does not make recommendations about the appropriateness of particular techniques to specific styles of user interface.

The catalogue is organised into seven sections, the first of which is of a general nature, the remaining six corresponding to the principal application areas at which the MG-1 is likely to be directed, namely general control techniques; CAD and draughting; text manipulation; modelling and data presentation; image manipulation; and animation.

8.3.2. Control Techniques.

Certain techniques apply across a range of different applications. There are two types:

(1) techniques such as object selection, which achieve a similar effect within each application, but which operate in different ways; and

(2) techniques such as pop-up menus that can be used in a wide variety of specific applications.

The current section is concerned with techniques of the second variety. Techniques of the first type are discussed under the appropriate headings.

8.3.2.1. Type-in Box.

The type-in box is a rectangular area of the screen used for text entry. The user may be required to begin by placing a "caret" in the box, and then entering text and undertaking various editing activities. Support of type-ahead (where keystrokes occur faster than echoing) is a general requirement.

8.3.2.2. Cut-and-paste Buffer.

This is a buffer for use in editing operations. A CUT command removes the selected text into the buffer; a PASTE command inserts the buffer contents at the selected position.

Note that in the case of applications dealing with multiple data types, such as text-graphics editors, the buffer must be capable of handling the full range of data types.

8.3.2.3. Icons.

Icons are miniature graphical representations of the application's objects and services. Icons representing objects are used as a means of on-screen filing. Icons representing services are used for invoking commands, sometimes just like function keys, sometimes by moving an object-icon over the service-icon.

An icon may be created in a number of ways, for instance by setting aside the contents of a window, by creating a new document or file, or by duplicating the object represented by the icon.

Individual icons can normally be dragged around the screen. Most systems supporting icons also provide a means of arranging icons neatly on the screen, invoked either manually or automatically. In some cases, the system animates the rearrangement to highlight the repositioning of each icon.

8.3.2.4. Split-screen Controls.

A split-screen system allows a screen or window to be divided into two or more regions by means of horizontal dividers. Some applications support vertical split-screen dividers. The divider may be a simple horizontal line or may actually contain information about the region created.

Splitting is usually achieved by indicating the divider's position and giving a SPLIT command. The divider is removed by indicating which region is to go away and giving a DELETE command. The user may split a window by dragging a divider symbol to the required position, and reverse the process by dragging the divider back to its original position.

8.3.2.5. Tiled Windows.

A tiled window scheme can be implemented by allowing several levels of splitting. In this way, a region created by means of horizontal dividers can itself be split up with vertical dividers. Tiled windows require the same controls as split screens.

8.3.2.6. Overlapping Windows.

Rectangular overlapping windows require four types of controls:

(1) Window movement. A particular portion of the window boundary serves as the handle for window movement; the user points here and drags the window to its new position. During dragging, feedback shows the window outline. Because windows can be moved partly offscreen, a single small handle area is inadequate and so the handle symbols should be provided in all four corners of the window. Several systems use the title bar, although this prevents the user from moving windows offscreen upwards.

(2) Size changing. This again requires handles. While some systems have a single handle in the bottom right-hand corner, the standard MG-1 window control system, for example, provides size control symbols for each of the window's four sides. Window contents should remain fixed relative to the top left corner during size changes.

(3) Changing priority. Priority indicates which window is to be on top of the stack. Usually this is done by pointing at a "priority handle". Some systems require the selected window to be highest in priority; selection automatically brings the window to the top. A few systems allow other priority changes, for example sending a window to the bottom.

(4) Scrolling. This operation is required independently of the windowing scheme, and is discussed separately in section 8.3.2.7.

Windows require labels or icons for the handles, and also require a descriptive label for the window contents. Some applications attach command menus to each window; see section 8.3.2.8.

8.3.2.7. Scrolling.

Scrolling is required in all applications whose data extends over a larger area than is visible on the screen. This is often the case with text editors, spreadsheets, and circuit design.

Some applications require scrolling in only one dimension. It may then be possible to implement scrolling more simply. However, a great many different systems have been implemented for scrolling. Some make use of scrolling keys, while others use Control or Escape sequences of varying clarity.

In general, scrolling operations need to specify the axis of movement. Accordingly, they are often implemented via handles along the appropriate edge of the screen or window. Handles may be provided for scrolling by a single text-line (or equivalent unit in non-text applications), by a full or half screenful, or by a specified distance. It should be possible to invoke these functions several times in quick succession by "click-ahead", and have the window contents move only once throughout the total required distance.

Feedback is required to support scrolling; it should show the window's position relative to the overall displayed area. This may be done by means of an "elevator", a box that moves along the screen edge as the contents scroll. In CAD and draughting it is common to provide two or more views of the same drawing; see section 8.3.3.10.

A thumbing control allows the user to specify a position relative to the start and end of the document, and the window is scrolled to show that position. This may be done by the user dragging the elevator to the required position.

8.3.2.8. Fixed Menus.

Command menus usually consist of items of text displayed in a horizontal row or vertical column. Pictorial items may be used instead, where space is tight or the meaning of the pictures is obvious. To invoke a command, the user points at the appropriate item with the mouse. Usually the mouse-button downstroke simply generates feedback, confirming which item has been selected; the upstroke completes the selection. This allows the user to move off the item before releasing the button.

Menus of this kind may be arranged along any edge of the screen, although for the benefit of right-handers the left edge is usually avoided; also the bottom edge is sometimes preferred to the top because it is a more natural resting place for the user's gaze.

Menus may be fixed to the borders of windows. This provides a useful way of avoiding operand selection; the command implicitly applies to the window contents. Window-border menus are almost always placed at the top or bottom to reduce the amount of border width required. Scrolling controls are required for menus if their windows are adjustable in width.

Many applications require more commands than will fit in a single menu. Commands are therefore divided into a number of sub-menus brought up by selecting a command in the main menu. Many systems use this method, thereby extending the command repertoire to many levels of hierarchy. They use the ESC key for example, for backing up the menu tree.

8.3.2.9. Pop-up Menus.

Pop-up menus are temporary menus that are displayed when needed by the user, and later removed. The advantages are that they avoid cluttering the screen when not in use, and that they reduce hand movement by appearing near the mouse cursor.

The "standard" pop-up menu operates as follows. It appears, close to the cursor, when a particular mouse button is pressed. The user slides the cursor, with the button still pressed down, to the menu item required; feedback shows the currently selected item. The user then releases the button: the menu disappears and the requested action is taken. Next time the menu is required, the system will often display it with the most recently selected item under the cursor, on the assumption that this item will be selected again.

Some pop-ups stay on the screen until they are explicitly removed by the user. This is done for various reasons, such as allowing the user to scroll a very large menu (see below), or to allow several different parameters to be set. It is possible with these "stay-up" menus to accumulate several on the screen at one time.

Pop-up menus can contain a very large number of entries. To avoid obliterating large areas of the screen the following techniques can be used:

(1) Scrolling pop-ups. This technique is often used in menu managers, where the menus are in fact stay-ups. It is of course possible to scroll a pop-up without releasing the menu button.

(2) Overlapping pop-ups. These are displayed as a closely-packed group with a small portion of each menu visible, perhaps just enough to show a title. As the cursor moves into the visible part, the menu pops to top priority.

(3) 2D pop-ups. These may include items with right-pointing arrows. The user can slide off to the right, whereupon another menu pops up, a submenu of the previous one. The user can retract a submenu invocation by moving off to the left. This scheme allows several levels of sub-menus.

Note that the structure of overlapping and 2D pop-ups makes them equivalent to submenu schemes. Scrolling menus are basically one-dimensional in structure.

8.3.2.10. Pull-down Menus.

Pull-downs are effectively pop-up sub-menus applied to fixed main menus. The submenu pops up when the user selects a main-menu item; it then behaves just like a normal pop-up; the user slides down to an item (with feedback showing what is selected) and releases, whereupon the pull-down "rolls back up" and disappears, and the selected action is executed. This method is very popular; a similar method uses pull-downs within a main menu displayed along the side of the screen; to make room for the submenu, the lower portion of the main menu slides gracefully downwards out of the way.

8.3.2.11. Filing and Retrieval Techniques.

Under GENIX, most filing and retrieval is performed with the aid of path names. Several techniques have been developed to alleviate users' problems with path names, for instance forgetting the path, or mis-spelling the name.

Automatic name completion can be invoked when the user types a special key, for example, Escape. The system attempts to complete the name from the first few characters typed.

Iconic or spatial file indexes show a directory's contents in tabular or tree-structured form, and allow the user to select a file by pointing. Various methods permit new files to be created, for example, typing a name into the table, or labelling a spare branch of the tree.

It is possible to do away with file names altogether in spatial indexes, relying entirely on an object-oriented approach using non-unique free-form text descriptions. This method may have many side-effects, especially where it is used within a multiple-window desktop environment. It then becomes necessary to keep track, in the file index, of the files that are currently displayed in windows. When text descriptions are edited, the changes must appear everywhere they are displayed; the description may be visible in more than one place at once.

8.3.2.12. Command Undo.

Applications are increasingly expected to provide UNDO facilities. These are difficult to provide in certain cases, for example where the object of the command is ambiguous, as in many text-editing operations; or where buffering of the previous state is difficult, as in on-screen painting.

Special Undo requirements are considered below under the headings of the relevant application areas.

8.3.2.13. Help and Warning.

Help and warning messages may be displayed either in a reserved area of the screen, or in a specially created window. Special windows are preferred because they can be bigger and are more likely to be noticed by the user. The area or window must frequently include a menu of options for the user, such as "ignore warning" or "cancel operation". The application must take appropriate action if the user ignores the warning altogether and starts issuing more commands.

8.3.2.14. Cursor changes.

Most systems change the cursor to reflect interpretation of mouse buttons or cursor movement. This often happens when the mouse moves from one area of the screen to another, for example into a menu, into editable text, or onto a window "handle".

In some instances the cursor may disappear altogether. This may be appropriate when text is being entered, for example, when it is necessary to avoid clutter from a superimposed cursor and caret.

8.3.3. CAD and Draughting Techniques.

This section covers techniques concerned with the creation of two-dimensional geometric drawings. The two application areas in which these techniques are most widely used are CAD (especially circuit design) and the preparation of diagrams. Applications of this kind treat the screen rather like a drawing board; due to the screen's small size, the effect is often of a screen-size aperture onto a very large board.

Many of the techniques used in CAD and draughting have been carried over from line-drawing graphics. Some CAD packages are designed to run on either line-drawing or raster graphics equipment. The overall result is to treat the raster display like a line-drawing display; this may, however, create some problems in achieving an adequate level of performance.

8.3.3.1. Endpoint Placement.

Where geometric objects are being defined, the user must specify two or more endpoints. This is usually done with a click of a mouse button at each endpoint. The program should display a small cross or other identifying symbol; if written into the bitmap with XOR logic, symbols are easier to remove when the object is drawn or when the user retracts an endpoint.

Endpoint constraints are usually needed to ensure that points fall on grid intersections. However the user may wish to turn this constraint on and off.

8.3.3.2. Object Selection.

The user needs to be able to select geometric objects in order to move, copy, delete and modify geometry. Complete objects are selected by pointing somewhere on one of the lines making up the object. Often, a method for selecting an individual point on the object (for example, a line end), is required; this must not conflict with methods of selecting the whole object.

Deselection is usually achieved by selecting something else, if necessary by pointing into an empty area of the screen.

When an object has been selected, feedback is required. This can be provided either by showing the vertices of the object, or by flashing the object continuously off and on. It is sometimes effective to invert all the pixels within the object's surrounding rectangle, although this does not work well where objects overlap extensively. Feedback should be easily removeable when the object is deselected.

Endpoint selection should take precedence over object selection, so that when the cursor is within a certain distance of an endpoint it should "snap" onto it rather than onto the object as a whole. This is known as "gravity" field selection.

It is often necessary to select several objects at once: multiple selection can be supported in several ways. A simple way is for the user to draw a "rubber-band box" around all the objects to be selected; this area multiple selection is useable only if the objects are suitably positioned. The concept of the rubber-band box is covered in section 8.3.3.4.

Another approach is through the use of an EXTEND key or button. The Shift key may be used for extension; a shifted selection results in appending the selected object to the set of objects selected so far. This technique can be interleaved with scrolling operations in order to select objects scattered over a wide area.

8.3.3.3. Creating and Selecting Groups.

In some applications, the user can select several items and form them permanently into a group. Any action applicable to single selected objects can then be applied to an entire group. Several groups can themselves be formed into a group, and so on.

The user should be able to select entire groups as well as individual group members, down to the level of specific endpoints. One way to achieve this is to use successive "up-level" commands (for example, mouse-button clicks). Another way is to identify the scope of the command, such as point, object, group, before trying to select.

8.3.3.4. Rubber-band Operations.

Rubber-band techniques are used extensively in draughting operations. The two main reasons are:

(1) to explain to the user what is happening, for example to show that two points are being connected; and

(2) to provide feedback regarding the precise line or area being created, as in the case of area multiple selection.

Rubber-band drawing can be applied to straight lines, circles and arcs, rectangular boxes, constrained lines (horizontal, vertical or 45°), arrows, or any other objects. The user indicates the type of object, and presses down on a mouse button, lifting up when the object is correctly positioned. Numerical dimensions may be shown to assist in creating an object of the desired size.

A problem arises in drawing connected sequences of lines with rubber-band feedback. The normal technique would be to press down a mouse button to start each segment and release to finish it, pressing down to start another. However this leaves the user, at the end of the sequence, with a dangling rubber-band line that must be removed. Some systems avoid this problem by requiring a down-up at each endpoint, instead of an up-down; thus the line segment itself is drawn with the button up, which is rather unconventional. Possibly a more acceptable solution is to have the user terminate the sequence with an extra click.

Use of the rubber-band technique is limited by the amount of computation required. Some programs provide a rubber-band feedback during scaling operations, but this can be used only on simple objects. The MG-1's GENIX provides an in-built facility for the fast echoing of rubber-band

lines.

8.3.3.5. Moving and Copying.

When an object has been selected it can be moved or copied to a new position. Often the two actions, moving and copying, share the same syntax. Copying may be carried out by duplicating, which creates a copy alongside the original; a MOVE command is then used to position the copy correctly.

Moving an item is quite straightforward provided the distance to be moved can be specified unambiguously. This is the case if:

(1) the item, and its destination, are single points, as with moving an endpoint; or

(2) the new position is specified by a relative distance.

To specify the new position of a complex object, either the user must indicate a reference point on the object, which is equivalent to (2) above, or the system must provide feedback in the form of dragging.

When just one endpoint is to be moved, the system may provide rubber-band feedback. All lines connecting to the point must be shown. If they were originally constrained, for example, to horizontal or vertical, they must remain so.

It may be necessary, in moving an object, to align it with another object. This must override any grid constraint currently in effect. As in endpoint placement, it should be possible to switch off the grid and let point-proximity influence positioning.

Moving and copying via cut-and-paste is rarely found in CAD or draughting applications because it is very difficult to apply accurately.

8.3.3.6. Clean-up after Editing.

When two lines cross, and one is deleted, a gap may be left in the other. Some systems automatically check for such effects and repair the gaps: where this is not done, a "clean-up" command is needed. Such commands typically blank out the screen or window, and regenerate the whole display.

8.3.3.7. Scaling.

Objects may be scaled in x, y or both simultaneously. Scaling may be applied with or without dynamic (rubber-band) feedback. A useful case is scaling by -1, to produce mirror-imaging; this technique is described in section 8.2.9.

Scaling is performed with respect to an origin or baseline. This may be implicit, where, for example, the origin is assumed to be at the centre point or top-left corner of the object. The user scales the object and then adjusts its position.

In other systems the user can specify a reference point, as in relative moves. For example, the user can specify two points in the current selected object and the two corresponding positions they should occupy after scaling; or a source and destination rectangle. Through rubber-band feedback, the source and destination can be maintained in alignment to avoid introducing a rotation. It is of course quite feasible to introduce rotation with scaling: three points in the source and three in the destination will specify a complete six-point transformation.

8.3.3.8. Rotation.

Rotation is usually specified in terms of an axis (centre of rotation) and an angle, as is described in section 8.2.9. Some systems permit rotation only through multiples of 90°. As with scaling, rotation may be implemented in terms of a default axis, such as the centre of the object's bounding rectangle. The user then merely specifies the angle, and fixes up the position afterwards.

Rotation can be specified in a mode-free, down-up syntax through the provision of dynamic feedback. The user depresses a mouse button, with the cursor at the centre of rotation, and releases at a position in the required angular direction from the centre, relative to the appropriate positive axis. Dynamic feedback shows the rotated object as the cursor moves.

As mentioned in the previous section, rotation can be achieved by specifying a six-point transformation. However this is an awkward method of applying pure rotation without scaling.

When entire groups are rotated, the effect is combined with any transformations previously applied to individual objects. It may be difficult, as a result, to modify an individual transformation after rotating a group, for example, to change the x-scaling of a box.

8.3.3.9. Curve Editing.

Curves are usually specified in terms of control points along their length. In the popular Bezier and B-spline techniques, the control points are not on the curve itself but on a polygon that follows roughly the same path as the curve. This polygon may be created and edited in much the same way as ordinary straight-line sequences. After each edit, the curve is recomputed and displayed. If sufficient computing resources are available, the curve can be recomputed dynamically as control points are moved.

Some curve-manipulation techniques, for example, Beta-splines, have the control points on the curve itself. The user then effectively points at the place requiring editing, and repositions the selected point.

8.3.3.10. Multiple Simultaneous Views.

The user often wishes to see two or more views of the same information simultaneously. For example, when working at high magnification, the user may want to see the whole drawing from time to time. This effect is usually supported through split screens or multiple windows. The user should be able to work in either such window.

8.3.3.11. Undoing Geometric Operations.

Geometric editing operations are usually straightforward to undo because:

(1) the inverse of any operation is itself a geometric operation. For example, scaling up undoes scaling down, and an insert undoes a deletion;

(2) the amount of data to be retained for Undo purposes is small.

8.3.4. Text Manipulation.

Text manipulation involves a very extensive range of techniques. In terms of interaction, however, these can be reduced to a relatively small number of generic operations: text entry; changing character attributes; changing paragraph attributes; page layout; document traversal; and a number of miscellaneous operations such as command reversal.

The degree of interaction varies. Most of these operations are applied single-step; the user changes a parameter, and the system responds with a new display of the document. This kind of operation is discussed only briefly here as the emphasis is on techniques requiring rapid response or dynamic feedback.

8.3.4.1. Text Entry.

Text is normally entered in place; the user places a caret and types. Line breaks should appear automatically. The system should recognise hyphens as permissible places for line breaks.

In modern systems, paragraphs are reformatted automatically as text is entered in the middle. This is not a simple extension. For example, if a short word is entered at the beginning of a line, it may need to be positioned at the end of the preceding line. Thus it is inadequate to reformat just from the current line onwards.

As text is entered, the system should be capable of making changes to the current font, and should provide commands for altering the style of specific strings of text, for example, bold and italic face, subscripts and superscripts, underlining, and as many other effects as possible. This may be reflected in a change to the cursor shape.

8.3.4.2. Text Selection.

To select a caret position the user points and clicks with the mouse.

A sequence of text may be selected by drawing across it, that is by positioning at one end and moving to the other end with the mouse button down. An alternative and sometimes easier method is extension of the selection: after the start has been selected, the end is specified with a different mouse button or by using a Shift key. This method was introduced in section 8.3.3.2.

The selection may also be extended up the text hierarchy, to encompass the current word, line, paragraph, page or entire document. This is sometimes done with multiple mouse clicks.

Text selection feedback is provided through inversion, outlining or underlining. It should not be applied to white space to the left or right of the paragraph.

8.3.4.3. Changes in Format.

Changes in format can be made at the paragraph and page level. Paragraph formatting is controlled partly by the tab rack and partly by paragraph properties that include alignment rules and settings for lines and paragraphs. They are changed by modifying the settings in a tab-rack display or property sheet; the latter is typically displayed in a window. Pull-down menus may be used for property changes.

Page layout is also usually changed with the aid of property sheets. The user may be allowed to draw the required layout, including multiple columns, on the screen.

In some systems, feedback is provided to show the type of option displayed, such as normal or indented paragraphs, headings, and footnotes, as in a typesetter's stylesheet. Other methods of feedback can be devised, and are useful where the type of entity is easily misunderstood.

8.3.4.4. Graphics Text.

The term "graphics text" refers to text labels in graphical material, such as found in illustrations, charts, and circuit designs. Graphics text requires its own techniques for manipulation.

Single-line items of graphics text can be entered by first specifying a position and then typing. The position may be either the left-hand end of the text baseline, the right-hand end, or the centre; this option is specified when the text-insert command is given. During type-in, changes in font and style can be made as with normal text entry.

Text items, once entered, can be manipulated in various ways. Text copying and movement should be available: it may be desirable to show block outlines during movement, rather than the full text. Some systems also allow rotation through multiples of 90°. Arbitrary rotation is rare, and users do not expect to be able to edit text in a rotated position.

In more complex systems, multi-line text items can be added to drawings. An enclosing box should be defined, and a subset of normal word-processing functions, such as paragraph formatting and justification, should be available.

8.3.5. Modelling Techniques.

The term modelling applies to applications in which an abstract model is maintained, and the display shows a view of the model that permits certain forms of editing and simulation. Examples include: 3D object design; molecular modelling; the viewing of scientific simulations; and network modelling and simulation.

As far as interaction is concerned, operations on models fall into three categories:

(1) Editing the model through graphical or typed-in commands;

(2) Regenerating the model display to show the effects of changes; and

(3) Simulating continuous phenomena, for example, nuclear reactor behaviour and rotating molecules.

Given the very wide variety of computer models, it is not feasible to describe the full range of techniques applicable. A few examples are given here.

8.3.5.1. Editing the Model

Editing 3D objects generally requires multiple views. The user works in one view, and can see the corresponding point in the other views.

Network editing involves changing the nodes and arcs in the network diagram, and editing text labels. A popular method is to allow free-form editing, followed by a semantic check. Other systems apply comprehensive validity checks to every operation.

8.3.5.2. Regenerating the Display.

This is a major issue in 3D modelling, where display generation involves perspective transformations and the removal of hidden surfaces. Where ray-tracing algorithms are used for extra realism, computational loads are often heavy.

In simpler situations, display regeneration is required to clean up after a session of model editing.

8.3.5.3. Continuous Process Simulation.

These applications can be categorised into real-time and compute-bound processes.

Real-time applications are typified by aircraft simulation, where a fresh image is required every refresh cycle. In some cases, such as video games, only partial image update is required.

Compute-bound processes include most cases of physical process simulation, as with nuclear reactors, airframes, and traffic flow. Many of these applications are designed for only partial image update.

8.3.6. Image Manipulation Techniques.

The black-and-white display is not a particularly versatile image-manipulation device, due to its low gray-scale resolution. However, some applications have been developed for image editing, and are useful in specialised circumstances, such as preparing mockups of screen layouts. A number of painting techniques have been developed for applications of this kind.

8.3.6.1. Painting.

Freehand painting involves combining the pixels of an arbitrary brush with the pixels of the screen image. OR functions are used for adding black pixels, and NOR for erasing. Some systems implement textured painting and airbrushing.

Constrained painting deposits pixels in the "cracks" of the grid. Extra care is needed at corners.

Enlarged painting, working on so-called "fat bits", is useful for detailed work.

8.3.6.2. Image Editing.

Image editing techniques are used to crop images, move and copy image sections, read images from file, and to perform other essential operations. Where images are to be repositioned, dragging is useful but expensive.

Techniques for scaling and rotating images are often useful. They depend on sampling algorithms (for arbitrary scale factors and angles) and rapid bit manipulation (for integer scale factors and multiples of 90°).

8.3.6.3. Undoing Image Manipulations.

Operation reversal can be difficult or expensive to provide because large areas of the edited image may have to be saved before editing begins.

8.3.7. Animation Techniques.

These techniques, for the most part, involve rapid regeneration of the screen image, although some techniques are based on only partial regeneration. In general, therefore, animation techniques make similar demands to techniques for simulating continuous processes.

8.4. Device Independent Graphics Systems.

The objective of a standardised graphics package is to overcome the problems of program and programmer portability. Such packages aim to be as device-independent as possible: while this is fairly easy to achieve on the level of specific I/O devices supported by a specific computer, it is much more difficult when the objective is machine-independence. However, certain of these packages have made great advances in standardising system predictability.

Standard packages are built around two levels: dependence and independence. The device-dependent internal structure is provided with a user interface that attempts to provide a standard array of facilities.

A major result of this two-level approach is modularity. The package as a whole can be built as a series of self-contained but mutually supporting modules. The MG-1 provides a specific set of routines for data input and the creation and manipulation of pictures. The machine specific procedures provide a general modelling capability that are the basis of a higher level package. The package then provides a wide array of modelling applications from that general capability.

For example, a single machine need not be provided with an exhaustive array of text styles, from crude dot matrix to high quality phototypeset. Instead, a general text-handling facility can be provided, and a facilities provided for the conversion of a standard text type into different quality output.

Another useful product of this duality is the ability of a specific system to apply non-standardised world coordinate transformations to a picture, and then feed the result into the display space system in a predictable fashion. This is a further instance of a modular approach limiting the range of functions that need be physically implemented while not precluding sophisticated effects.

A major design consideration of such a system is the command language used to implement these facilities. Available systems typically use a high-level language such as FORTRAN or Pascal to provide a set of device-independent functions. Thus, each standard graphics system will use a specific language binding to implement a full range of functions for primitive handling, transformations, input, and systems control. Problems of inter-machine portability are therefore simply a matter of dialect differences: a single programmer with only a limited range of specialist knowledge can quite easily handle such problems as may arise.

A number of such standardised graphics systems are available. The GKS (Graphical Kernel System) implements the idea of modularity to the extent of making available a graded package of facilities. Because the whole package is highly comprehensive, and not every installation will need the complete range of features, nine upwardly-compatible levels are available. The MG-1 implements the eighth level, 2b.

In all cases, input and output control routines are supplied. The special hardware and software features of the MG-1 are employed via the Generalised Drawing Primitive, which, while relating to MG-1 specific features, does so in a predictable fashion. All transformations are handled in application specific coordinates, and moved into device space for display. This sequence of events forms a "graphics pipeline", taking the form:

> segment store > transform segment > clip > transform to device coordinates > output to device.

This is a standard pattern. The MG-1's Window Manager has the effect of modifying the above pipeline, but also in a standardised and well-understood fashion:

> segment store > transform segment > clip > transform to device coordinates > output to window > map the window to physical screen bit map.

The last two stages of the modified pipeline are beyond the control and knowledge of the graphics package. Instead, output is fed to the Window Manager, and the MG-1 executes its normal operations.

At the heart of the GKS standardisation of machine-specific implementations is the concept of the Work Station. A Work Station can be implemented by a variety of hardware and software. The actual mixture is immaterial because the workstation concept is an abstract one, and merely specifies a minimum range of facilities. If these facilities are not directly implemented in hardware, they must be simulated in software. So long as the minimum range is available, the package will operate. This is the case with the MG-1.

8.5. The Window Manager

8.5.1. Introduction.

The window system consists of four levels of software. The first level is that of the basic graphics primitives and the rasterops. These have been described in some detail in section 8.1, and are responsible for the creation of the images.

8.5.1.1. The Panellist Element

The panellist is a set of extensions to the kernel, and is responsible for maintaining the screen image. The data that is displayed in the screen image is stored in memory in raster form, and is mapped onto the panel. This mapping process is the same as that described in section 3.2.1.3 because the raster is mapped into the screen bit map in page form. Panels supplied in this way are known as "buffered panels".

"Physical panels" are those where the screen image is taken directly from the user's bit map, and which thereby operate almost instantaneously.

Where rasters undergo some change, a buffered panel is updated using the PanelUpdate function. Until this function is called, the screen image cannot be altered.

In order to improve the performance of image updating, only the altered areas are updated, while the rest of the raster is left unchanged. The changed areas of the raster are specified with a parameter to PanelUpdate.

In the case of Physical Panels, the PanelUpdate is a null operation, because the panels are refreshed directly from the user memory, they must use contiguous memory locations for consecutive scan lines, and are therefore restricted to full screen width. Because of their relationship with physical address space, they may be moved vertically on the screen only in multiples of four scan lines, and are fixed horizontally. However, these disadvantages may be outweighed by the speed of image refresh.

8.5.1.2. The Window Manager Element

The Window Manager controls the contents and behaviour of the panels as they appear on the screen, and the combination of the various panels needed to create a fully interactive window. Furthermore, the window manager stores the positions, sizes, and priorities of each window and associated icon for resumption at the next login.

Panels are organised in a tree-structured hierarchy, and are combined to build up successively more aggregated entities. Each window is made up of a subsidiary interior panel for the display of process data, and four border panels which contain the control symbols and a name field. A window is established by the Window manager whenever one of the window creation calls is made. These are described in section 8.5.5.

8.5.1.3. The Tool Library Element

The tools available to the application-writer are supplied by the Tool Library in such a form as to encourage standardisation. They include pop-up menus and scrolling systems. These are considered in section 8.3.

These four software elements combine in order to provide a management system that can control one or more windows simultaneously. Each is owned by a process such as a text editor or a compiler, and is active until the completion of the invoking command. Because each window is controlled by a process in memory that behaves as though it had sole command of the system, each window is a "virtual terminal".

Windows consist of a rectangular screen area made up of a rectangular display area and four borders. Within the borders are a name field to identify the controlling process, and a series of menu options. These are selected by clicked one of the mouse buttons over the appropriate symbol, and are used to execute such routines as window sub-division, movement, and priority control.

8.5.2. Window Control.

The Window Manager provides two types of control. Each individual window has its own standardised controls for movement, size changes, and priority control. These are the same for all windows. In addition are a number of controls such as pop-up menus that are required by individual applications, but administered by the Window Manager. Pop-up menus are obtained by clicking or pressing a mouse button over the appropriate area of the screen. Option selection is also a function of mouse button clicks, in conjunction with the mouse-controlled cursor.

8.5.3. Cursor Behaviour.

The cursor provides another element to the window system. The screen has its own cursor which follows the mouse, and is used to move around the whole work area. Windows receiving keyboard input have their own additional cursors or carets.

Upto sixteen cursor images are available to each process. These are selected according to the area of the screen, and mixed into the display. As the cursor is moved by the mouse, it changes from one image to another at a rate undetectable by the user. Some special applications such as painting and sketching packages, may have a very large number of cursors, of different sizes and shapes.

8.5.4. Input Control.

Input from the mouse and keyboard are channelled into the appropriate process via the PanelRead routine. Input events are passed through a Panel Mask which acts as a filter. The appropriate filtering conditions are set using the PanelSetMask function.

The filtering routines of the panel mask create an input event queue: keyboard events are channelled into the window currently selected for keyboard input, while mouse events are channelled into the window currently under the mouse. The effect of the queuing system is to provide each recipient with a logical and acceptable instruction stream.

The Panellist contains an emulator of the VT100 terminal with VT200 enhancements. Using this, any MG-1 key can be re-programmed to generate a new character or set of characters. Whole command strings can be generated with one key in this way, without interfering with the normal ASCII code assignments. The routine used is described in section 5.8.1 of this Guide, and in the accompanying Window Manager documentation.

8.5.5. Creating Windows.

The Window Manager will provide a new window on receipt of the WindowCreate call. The resulting window or its icon will not become visible on the display until the WindowStow or WindowUnstow calls are supplied. The WindowCreate call is frequently too low level, and so functions are provided for the creation of different types of window. In all cases, the full window structure is provided, and a name given.

Three types of window are available; full-function, physical panel, and TTY. To create a full-function or physical panel window, the WinCreate function is used: for a tty window, the TTYWinCreate call is used. In both cases, the arguments to these calls are described in the Tool Library section of the Window Manager documentation that accompanies this Guide. After the completion of these procedures, the window can be made visisble with the WinDisplay function.

8.5.6. A General Viewing Capability.

The Window Manager's viewing capability involves the four levels of software introduced above. The graphics routines and the rasterops are used to create images composed of lines, points, text characters, and solid areas. A mapping routine is used to transfer the memory representation of the image onto the relevent panel. The PanelUpdate function is responsible for the updating of screen images from the panel's screen raster. Because only a small proportion of the raster may change, the PanelUpdate function copies across only the altered data.

The Panellist controls the combination of display panel and border panels that make up a complete window as it appears on the screen. All input events, from mouse or keyboard, are handled by the panellist, and the Window Manager executes the appropriate command.

8.6. Interactive Applications.

For any complex graphics system, whether for the programmer or for the end-user, the most visible aspect is the user-interface. Three factors are involved in the creation of a good user interface: abstractness, predictability, and standardisation.

The precise nature of the user interface is regulated by the degree of abstractness involved. This determines the range of operations available without explaining how they are implemented. The programmer must know what is available without necessarily knowing how it is implemented. In order to ensure that the facilities available are as easy to use as possible, they must operate in as predictable a way as possible. This predictability is a function of standardisation.

The MG-1's graphics handling system is standardised on many levels. The high-level graphics packages available relate to international conventions, while providing access to the specific capabilities of individual machines; standard command languages are used throughout, for example, the generalised assignment of commands to the mouse buttons to avoid clashing with application command allocations; the MG-1's graphics library provides a set of primitives handling devices that acts as the basis of higher level operations, and are written in standard control languages; the command set operates in as normal a way as possible, and avoids accessing routines through unfamiliar combinations of key-strokes or other events.

A further element of system predictability is the use of feedback to highlight system states.

While this set of considerations is a major factor in determining the form of graphics systems for the application programmer, it should also influence the programmer during the writing of systems for the end-user. It is typically the case that an end-user will be less familiar with a system than the writer, unless consultation has been very close. The user interface should be very carefully designed; some of the references in Appendix F provide coverage of this specific point.

Chapter 9
System Administration

9.1. Introduction.

The MG-1 Workstation provides powerful computing facilities for both single-user and multi-user enviroments. As a personal workstation, the MG-1 may be used to reflect the idiosyncracies of individual user style. Additional management considerations must be taken into account if the MG-1 is to be used by two or more users.

This chapter outlines the major System Management considerations and should be read in conjunction with Chapter 10 which describes aspects of System Security for the MG-1.

This chapter will introduce a large number of powerful GENIX commands. The new user is advised to read the relevent sections in the GENIX Programmer's Manual: the experienced UNIX user should also refer to these sections to ensure familiarity with the more detailed syntactical conventions.

9.2. System Management and the Superuser.

The major responsibilities of System Management include:

1. the initial installation of the MG-1
2. the adding and deleting of user accounts, passwords and file systems
3. the monitoring of system resource distribution
4. the "backing up", or copying, of all files to guard against accidental loss of programs and data.

To protect confidentiality and guard against accidental data loss, GENIX automatically restricts access to many system files; for example, the rm command for file deletion will not access '..' system files. GENIX also provides a protection mechanism to allow users to restrict access to their own files.

There is one user, however, who has unlimited access to the system — the "superuser". This user automatically has the login name "root". A super user is required irrespective of whether the system is to be used in a single-user or multi-user environment.

The superuser can read and change anything on the system whether or not the material is protected against other users. Only the superuser can perform certain vital functions such as adding and deleting user acounts from the system.

There are three initial System Management considerations relating to the superuser:

1. Decide who is to be superuser. In a single-user environment the choice is already made. Within a multi-user environment there may be certain circumstances in which it is appropriate for more then one person to be able to perform the duties of superuser. Any person with knowledge of the root password can log in as a superuser.
2. With root's unlimited access even minor mistakes can cause major problems. Whoever performs the superuser duties should therefore carry out their everyday work as an ordinary user, and maintain a separate root account.

   Log in as root only when it's absolutely essential.
3. In a multi-user environment, ensure that the superuser facilities are available only to authorised users. In order to maintain system security, it is essential that passwords are known to as few personnel as possible.

9.3. Root Password.

The root password should not contain less than six characters nor should it be easy to guess: avoid proper names, nicknames, telephone numbers or obvious dictionary words. Preferably choose a random mixture of upper and lower case characters and numerals. Then be sure to commit the password to memory. Note that only the superuser can choose or change the root password.

After following the start-up procedures described in Chapter 5 the MG-1 screen will display the special root shell prompt. This is a '#' symbol, irrespective of the shell in use. In response to the login prompt, type

```
root
```

GENIX then acknowledges the command by displaying a further shell prompt. Now type the command

```
passwd root
```

The argument to 'passwd' is the user name, in this case 'root'. GENIX requests the new password and, as a further check, requests confirmation of the entry:

New password:
Retype new password:

Any discrepancy between the two entries causes an error message to warn of an entry mismatch.

To preserve security, the password is not "echoed" on the screen. After the password has been typed twice, the system automatically records the entry in the GENIX file /etc/passwd.

Pressing <RETURN> in response to the prompt for a new password represents a request not to change the existing password.

If a superuser forgets the root password, recovery is possible by way of the MG-1 Security Disk. This procedure is described in Chapter 11.

9.4. Adding new Users: newuser

Adding new accounts to the system is one of the tasks of a superuser. Even within a single-user environment, the superuser should follow the procedure described below in order to log in as an ordinary user.

Each user has a login name which is the name by which the user is known to the system. This is the name by which system surveys such as the 'who' command identify users, and accordingly, user names are public information.

Most users find it convenient to use a shortened form of their own real name, or perhaps their initials. Note that spaces, tabs and special characters, where used, are considered to be an integral part of the name and must be included at each login.

To assign a new user name, log in as root, and in response to the root prompt type:

```
# newuser
```

The system responds with:

Type login name:

The program now requests the full personal name of the new user. This is used in certain of the mailing options available under GENIX:

Type full name:

If the first part of the full name is the same as the login name, it can be represented by the ampersand (&) character. For example, a user called John Smith , with the user name 'john' could type:

```
& Smith
```

The system then procedes to establish the user account.

The new user account is now created and the various details stored in the GENIX /etc/passwd file. The new user can now log in on the system.

The superuser has no need to know other users' passwords since the superuser automatically has full access to the entire GENIX system, including user files that are protected against all other personnel.

Users can change their existing password by logging into the system, and against the shell prompt, typing

```
% passwd
```

GENIX responds by prompting once for the user's current password, if any, and twice for the user's choice of new password.

The password entry in the /etc/passwd file is changed to the new password. Root may occasionally need to alter entries in the /etc/passwd file; procedures for performing these changes are described in Volume II of the GENIX Programmer's Manual.

The superuser can change another user's password by issuing the following command:

```
% passwd username
```

and following the screen prompts as above.

9.5. Removing Users

Removing a user's access to the MG-1 system is a two stage process:

1. removing the user's files from the system, and
2. running an editor to remove the user's password from the password file.

User files should be carefully reviewed prior to deletion as they may be required later, and can be transferred to other users and to floppy disk. It is important to remember that user mail should be removed also.

To remove a user's files (using 'john' as a specimen user name) type:

```
% cd /users/john
% rm –fr *
% rm -f /usr/spool/mail/john
```

These three commands respectively change position to the relevant user directory, and force deletion of both files and directory, and remove the user's mail box.

Finally, an editor such as ed can be used to remove the user's password from the file /etc/passwd.

Type the login name of the user to be removed from the system, followed by <RETURN>. Deluser marks the user's entry in file /etc/passwd as unusable. If the user has mail, all messages are deleted, and the user's mailbox is removed from the system.

9.6. System Integrity

During day-to-day use of the system a number of inconsistencies can occur. For example a system crash can adversely affect the internal GENIX housekeeping systems which may not be able to perform their normal functions. This may result in directory listings not reflecting the actual file system, or there being file corruptions within the directory structure. The fsck program performs a file system check to resolve these inconsistencies.

9.7. The fsck Program.

The five-phase fsck program is run automatically by GENIX at startup time but the superuser may run it interactively at any time.

The fsck program should be run only while the system is in 'system maintenance' (single user) mode. This is to ensure that no user processes are using the system or accessing files.

The following command performs a check on file system h on disk device hd0:

```
% fsck /dev/hd0h
```

A report of the five checking phases of the program is displayed together with a report on the number of files, blocks used and available blocks on the system:

```
** Phase 1 – Check Blocks and Sizes
** Phase 2 – Check Pathnames
** Phase 3 – Check Connectivity
** Phase 4 – Check Reference Counts
** Phase 5 – Check Free List

1137 files  8387 blocks  2651 free
```

If fsck finds a problem then it reports to the user and asks if a correction is required. Occasionally a damaged or inconsistent file has to be deleted by GENIX in order to make the correction. In such cases, loss of information is usually minor and GENIX reports on the nature of the correction. Remember that any minor deletions which GENIX has to perform are invariably less damaging to the system than a file inconsistency which is allowed to continue.

For full details, see section 8 of Volume I of the GENIX Programmer's Manual

9.8. Daemon Processes

GENIX runs a series of programs which run automatically whenever the system is in use. Called "daemons" (pronounced "demons"), these programs run continually and periodically activate system checks and basic system functions.

This suite of programs includes:

1. The "update" daemon which activates the "sync" primitive on an automatic basis every thirty seconds. All data in core memory that should be stored on disk is written out to ensure that the contents of the system disk are as up to date as possible in the event of a major problem such as a system crash or abnormal shut-down.
2. The "lpd" daemon superintends operation of the line printer, /dev/lp. Baud rate is set as an argument to the daemon. Console-displayed error messages are available to diagnose lpd problems.
3. The "cron" daemon acts as an internal alarm clock for the control of commands and jobs to be executed at pre-specified times. The cron program repeatedly examines a file called /usr/lib/crontab for instructions to perform these functions.

9.9. Disk Space

Available disk space quickly becomes a valuable commodity on any computer system as users compile programs, create and edit files and directories, and perform other tasks. When the system runs out of disk space, it is effectively paralysed; no new files can be created, and existing files cannot expand.

A key function of the superuser is to monitor space availablility and usage. It is advisable to perform a periodic disk space inventory, especially in a multi-user environment, in order to monitor the use being made of disk space by individual users. The frequency of survey depends on the demands being made on the system.

GENIX provides three principal tools for monitoring disk space;

df monitors the amount of free disk blocks,

quot summarises each user's disk space quota, and

du summarises disk usage.

In the following discussion of these commands, only the general features are described. For full details of these commands refer to Volume I of the GENIX Programmer's Manual.

9.10. The df command.

The df command reports the number of free blocks available on a specified data structure. If the argument is a filesystem (eg /users/john), df reports the number of free blocks available on that filesystem: if the argument is a filename, the report displays the number of free blocks in the filesystem containing that file. If no filesystem is specified the available free space on all filesystems is displayed. The reported numbers are in double block units of 1024 bytes.

With experience, the superuser will become aware of the figure which represents a comfortable margin for operation of the system. As a general rule it is advisable to maintain available free space at around 15% of total capacity, more if system usage fluctuates, less if it is relitavely stable.

9.11. The quot command.

If, after using the df command, it becomes clear that a disk space problem exists, the quot command can be used to determine the number of blocks owned by each user.

For example:

```
% quot –f /dev/hd0a
```

produces a list of every file on the root file system together with the name of the file's owner and the file size in blocks.

9.12. The du command.

The disk usage command (du) reports the number of blocks (of 512 bytes) currently being used by individual directories, subdirectories and files.

If du is used without specifying the name of a file or directory, the size in blocks of everything below the current directory is reported. In order to increase or decrease the scope of the du operation, change directory position with the cd command.

If a directory is specified, du reports the block size of that directory. For example

```
% du /users
```

generates a complete report on disk utilisation since the /users directory contains all the subdirectories and files for all currently recognised users.

The following is an example du report produced in a multi-user environment:

```
108     /users/richard/admin
478     /users/richard/dgraphic
1099    /users/richard/junk
1685    /users/richard
36      /users/john/manual
235     /users/john/documentation
271     /users/john
```

As well as displaying the number of blocks owned by individual files, du gives a total of the blocks used by a particular user. The three files (admin, graphic and junk) held in the 'richard' directory add up to 1685 blocks.

Similarly, John's usage is 271 blocks. In this instance, the superuser would probably wish to discuss with Richard whether there was a genuine need for all the disk space currently in use.

9.13. File Systems and Archiving.

The formatted hard disk is the basic storage mechanism for the MG-1's operating system and user data systems.

These various data structures are stored as a tree-structured hierarchy of files known as the file system. The entire file system can reside permanently on the hard disk: however it is advisable to take advantage of the facilities for distributing data among a combination of devices.

Additional file systems can be appended, or "mounted", to specified directories within the hard disk's resident file system. Once an additional file system is mounted, the GENIX copy and move

commands (cp and mv) can be used to transfer files between physically separate storage media. Mounting a file system held on a transportable medium, such as a floppy disk, allows the creation of additional space on the integral hard disk, the physical transfer of information between remote sites and the addition of applications packages. These procedures are run through commands held in the /bin directory, and may be executed by the superuser or by normal users.

The overall procedure involves:

1. formatting one or more floppy disks,
2. providing them with an outline file structure,
3. connecting the physically separate file structures,
4. moving or copying the relevant files, and
5. unmounting the floppy disk files.

9.14. Formatting a Floppy Disk: fdfmt

To format a disk which includes a write protect tab, first ensure that the disk does not already contain valuable information, then remove the tab and insert the disk into the disk drive.

In response to the shell prompt type:

```
% fdfmt
```

The floppy disk drive light will illuminate until the formatting is complete, a process that usually takes approximately one minute.

A floppy disk already containing data may be reformatted using the same routine, but care must be taken to ensure that none of the information is needed, as reformatting completely erases existing files and directories.

9.15. Creating a Floppy Disk File System: mkfs

Insert a formatted floppy disk into the disk drive and type the mkfs command plus the appropriate arguments in response to the shell prompt, for example:

```
% mkfs /dev/rflp 800
```

The first argument to the command is the filename of the archive; /dev/rflp is the default. The second argument is a numeric size measured in kilobytes. The effect of this operation is to create a single empty directory on the floppy disk.

The floppy disk drive light will again illuminate, for about 15 seconds, by which time the file system will have been created.

9.16. Mounting a Floppy Disk File System: mount

Load the formatted floppy disk containing a file system into the disk drive. In response to the GENIX prompt, type the mount command plus the appropriate arguments, for example:

```
% mount /dev/flp /flp
```

The first argument announces which device contains the mounted file structure. In this case, the floppy disk drive (driven through the /dev directory) is the parent device. The second argument is the name of the directory created on the floppy disk by the mkfs command. In this case, the directory is called /flp.

Once mounted, the floppy disk file system becomes an integral part of the hard disk file structure and can be treated in the same way as any other directory.

9.17. File Transfer to Floppy Disk cp or mv

To copy a hard disk file onto a floppy disk insert the formatted floppy disk containing a file system into the disk drive and, in response to the shell prompt, type the cp command, for example:

```
% cp myprog.c /flp
```

The first argument is the filename, with pathname if necessary; the second argument is the destination device, in this case the floppy disk drive.

To move a file from the hard disk to the floppy disk use the move command mv:

```
% mv myprog.c /flp
```

Note that the file is physically removed from the source device by the move command and is no longer accessible on the hard disk.

9.18. File Transfer from Floppy Disk: cp or mv

To copy a file contained on a mounted floppy disk onto the hard disk, the above procedures are reversed:

```
% cp /flp/myprog.c myprog.c
```

or, using the '.' abbreviation to repeat the last-used filename:

```
% cp /flp/myprog.c .
```

The two command arguments are the source and destination datanames respectively. The cp command can be used for installing application packages.

The mv command is used for file transfer to the hard disk if the file is no longer required on the floppy disk.

In order to protect this information, the –r option may be used:

```
% cp –r /flp/myprog.c .
```

This version of the command allows the disk to be read only, and not amended.

9.19. Unmounting a Floppy Disk File System: umount

To detach the file system held on a floppy disk from the hard disk file system, prior to removal of the floppy disk, type:

```
% umount /dev/flp
```

where the argument is the device currently containing the mounted files.

As a further protection, it is advisable to apply a write-protect tab to the disk.

9.20. Archiving and Backup.

The long-term storage and retrieval of infrequently used files (called "archiving") is achieved with the 'flar' command. The duplication of all or part of the hard disk system to prevent irretrievable loss of files due to system failure is called a "backup", and is achieved by using the 'dump' command, which directs a file system to either the floppy disk drive or a tape streamer, if fitted. Retrieval of previously dumped files is achieved with 'restor'. All of these are described below.

When new floppy disks are to be used as the archiving or 'back-up' media, they must first be formatted using the fdfmt command but the file system utility mkfs is not required.

The reader is referred to sections 1 and 8 respectively of Volume I of the GENIX Programmer's Manual for full details of these archiving and file system commands.

9.21. Archiving to Floppy Disk: flar c

To archive one or more directories or files onto a formatted floppy disk, use the cd command to move to the directory containing the data objects involved, and type the flar command, using the relative form of file name; for example:

```
% cd /users/john
% flar c file1 file2 file3
```

The options available to the flar command are many and varied, and are described in full in section 8 of Volume I of the Programmer's Manual. The arguments supplied above are as follows:

The 'c' element is a single character function code which instructs GENIX to create a new archive, overwriting any material already stored. The single function code used here may be combined with a number of other codes, which act as function modifiers.

/dev/rflp is the default name of the archive device, but other devices may be specified.

The third argument is the set of file names to be archived.

The initial move directory command "cd pathname" may be incorporated into the flar command by preceeding the list of directories or files to be copied with "–C pathname".

```
% flar c –C/users/john file1 file2 file3
```

The message "change volume" will be displayed when a floppy disk becomes full. To continue the archiving procedure, remove and label the full disk and insert another formatted disk.

If there are no formatted disks available, new disks can be formatted immediately because flar can be asked to return to the shell prompt each time a disk is filled. Run the floppy disk formatting system, fdfmt. When the disk drive light extinguishes, indicating the completion of formatting, type "exit" when using the C Shell, or the <Ctrl><d> sequence when using the Bourne Shell, against the shell prompt and the archiving procedure resumes.

9.22. Restoring Archived files: flar x

To restore the complete contents of a floppy disk to the current directory, type:

```
% flar x
```

The flar x function extracts files from an existing archive. Any function applied to flar can be modified by adding other codes. In order to restrict the operation to a specific list of files or directories, the data names should be given as an argument to the command, each name separated by a space.

9.23. Scheduling Backups: dump

It is customary to maintain two forms of system backup: a periodic full backup in which everything on the system is copied and, in the intervals, incremental backups to record the changes made to the files since the last full backup.

Naturally, the precise backup schedule depends on the degree of activity of the system. Even with low levels of activity, however, it is advisable to schedule periodic backups at least once a month. The description of the dump command in section 8 of Volume I of the Programmer's Manual, includes a suggestion for scheduling backups which utilises the backup levels (0–9) available with the dump command.

For example, to dump the entire filestore (dump level 0) for a 45 Mbyte system type:

```
% dump 0u dev/rhd0a
% dump 0u dev/rhd0h
% dump 0u dev/rhd0g
```

Only the first two of these commands (for partitions a and h) are required for 10 Mbyte and 22 Mbyte systems.

The default filenames are /dev/hf0a for the source device (ie the hard disk drive), and /dev/rflp for the destination device (ie the floppy disk drive).

These default names may be overruled using the –f option which accepts a given filname instead. Full details of the dump command are given in section 8 of Volume I of the GENIX Programmer's Manual.

9.24. Restoring Dumped Files: restor

The restor system reads back file systems, or individual files, dumped to a separate storage medium. Restor may, on rare occasions read back a whole dump (this is one way to copy a file system from one physical structure to another) but is more often used to read back files selectively. This is achieved by way of the x function referring to files and directories named as arguments to

the command.

The files and directories held on a dump volume can be listed using the dumpdir command.

The command

```
% restor x /john/memos
```

will restore '/john/memos' from the dump volume (accessed by way of its default filename /dev/rflp) to the hard disk. The x option is used to specify individual files to be restored. Note that this operation will not replace the file in directory '/user/john': instead, it writes it into a file given a numeric name by the restor procedure.

The dumpdir and restor commands, and their full range of options, are described in full in section 8 of Volume I of the GENIX Programmer's Manual.

9.25. Monitoring Processes: ps

All functions running on the system, such as system and user programs, editing, compiling and data entry are referred to as "processes". Each user may have several processes running simultaneously. To check these, the user types the process status command, ps:

```
% ps
```

which causes GENIX to list the user's processes currently running.

A variant of the ps command

```
% ps –ax
```

gives a list of all processes currently being run, irrespective of owner. This command is a useful system survey facility, and is of considerable value to the superuser. As with the majority of GENIX commands, many alternative options are available: full details are available from section 1 of Volume I of the Programmer's Manual.

The following is typical output from a ps –ax command:

PID	TT	STAT	TIME	COMMAND
3	2	Z	4.05	emacs shexec.c
74	6	T	2.15	–sh
76	3	R	1.22	csh
84	5	R	0.07	ps –ax

PID — is the "process ID" — a number GENIX generates and uses to identify a process,

TT — indicates the number of the terminal from which the process is running,

STAT — is the "state" report for each process:

- I Idle
- R Running
- S Sleeping
- T Stopped
- Z Terminated (Zombie),

TIME — indicates the total amount of time the process has been running, and

COMMAND — is the command name of the process.

9.26. Communication in a Multi-User Environment.

There may be occasions when the superuser will need to communicate with one or more users on the system. Similarly, users may need to communicate with each other and with the superuser.

Such communication can be instantaneous (the 'write' and 'wall' commands) or can be delayed until the user next logs in to the system (the 'mail' command, and the message of the day facility).

9.27. Message of the Day.

A built-in message of the day file can be used by the superuser as a bulletin board to keep users informed. Impending downtime, device availability, disk shortage, meetings, scheduled maintenance and other matters are typically documented using this facility.

The superuser enters the information in the file /etc/motd by means of a text editor such as ed, using the command:

```
% ed etc/motd
```

to create the message text.

Each time a user logs in, the message of the day is automatically printed on the screen. File access permission for /etc/motd is generally set as read access for all users and write access for administrative personnel only.

9.28. Software Administration.

Further important administrative facilities available to MG-1 users relate to software development, and include the MAKE and SCCS utilities.

1. MAKE updates a target file if it depends on prerequisite files that have been updated since the last update of the target itself. Commands held in /makefile operate upon a list of dependencies listed in /Makefile. Note that these are different. Dependency lists consist of target files and their associated prerequisite files, and may involve data objects held on more than one storage device.
2. The SCCS Source Code Control System is a series of commands for controlling the changes made to files containing text or program source code. It keeps track of all changes to the contents of a file, and makes each version available to the user. Comments explaining changes may be recorded; different versions may be merged; file differences may be listed; and unwanted amendments may be discarded.

SCCS is supported by its own on-line aid system called SCCSHELP. To operate this facility, type the sccshelp command, using the subject title as an argument.

Chapter 10
Security

10.1. Introduction.

This chapter on system security necessarily duplicates some of the information in Chapter 9. However, the general areas of System Administration and Security are sufficiently separate to warrant a modified approach.

GENIX is an operating system which allows several users to be logged into the MG-1 at the same time. It supports a full range of timesharing and multi-user facilities. To keep track of users and their needs, a human systems manager is required. In a multi-user environment this task is assigned to a privileged user, known as the superuser whose user identification is 'root', and who has responsibility for system security.

10.2. The Superuser.

The superuser possesses privileges relating to overall control of the system. Many of the superuser's functions have been described in Chapter 9.

The superuser can override all user-set file protections and can access all the files and programs on the system, some of which may be confidential or potentially dangerous to the integrity of the system. The only exceptions are those files that have been processed by the 'crypt' encoding utility. However, most of these sensitive areas are protected from the user operating in a non-root environment.

Superuser privilege is password protected. Without knowledge of the root password (arbitrarily chosen by the superuser) the ordinary user cannot gain access to the full variety of GENIX utilities and special files which support the essential security functions.

The role of the system manager in a multi-user environment is to authorise users, issue individual passwords and user identification numbers, create file directories and take care of the many administrative needs that arise when numerous people use the same computer.

10.3. Passwords.

A password protected system such as GENIX prevents unauthorised access to the system by demanding a password, arbitrarily chosen by the authorised user, and ideally consisting of not less than six mixed case alphanumeric characters. Passwords, like any other information on a GENIX system are file-stored, but gaining access to the contents of the password file does not compromise the secrecy of the passwords as they are held in a 'scrambled' or encrypted form.

10.4. System Security.

System maintenance operations should be carried out only by the superuser. In a single-user environment, these are available only to the user's root account. In a multi-user environment, maintenance should be carried out only when the superuser is alone in using the system. Many maintenance routines can compromise standard DP processes, and cause system failure.

To provide a higher level of security for the MG-1 Workstation, GENIX enters multi-user mode upon power-up and requires the superuser password to enter the single-user mode needed for many maintenance routines.

The MG-1 can be powered up in single-user mode only by bootstrapping GENIX from the security floppy disk available from Whitechapel. This disk should be stored securely by the superuser and used in the event of the root password being forgotten. This procedure is described in Chapter 11.

The procedure for initially assigning the superuser password is described in Chapter 9.

10.5. User Security.

The superuser issues every user a 'login name' (also called the 'user-id') and, on password protected systems, an initial password. Both are composed of lower case characters only; when logging in, ensure that the shift lock key on the keyboard is not engaged. The login name is public information, and is used in many of the system survey routines available to the user. The password is known only to the user, and can be changed at will with the passwd command, so long as the original password is known.

The superuser also assigns a "user group number" to each user associated with the same project or administrative task.

10.6. File and Directory Access Permission.

GENIX includes a formal scheme of assigning access permission to files and directories. Under this scheme, each data object can be accorded three forms of access permission, read (r), write (w), and execute (x). Any of these access codes may be refused.

The meaning of 'read' permission depends on whether the data object is a file or a directory.

To read a file is to access the information held in the file. A user therefore requires read permission to use the cat command on a file, for example.

To read a directory, which contains not individual pieces of information but a group of files and perhaps other directories, is to examine its "table of contents". A user therefore requires read permission to use the ls command, for example.

'Write' permission on a file allows the user to modify its contents. Write permission on a directory allows the user to modify it by creating, removing and renaming files or sub-directories.

The meaning of 'execute' permission also has two different meanings. It is only used in relation to a file when that file is a program, in which case it gives the user permission to run the program. Execute permission on a directory permits the user to search the directory for a file to execute.

There are also three categories of users, for whom specific forms of access permission can be assigned: an individual user (u), a group (g) and others (o).

The 'individual user' is the owner of the file or directory. A 'group' comprises users sharing the same group ID number. The group ID 200 is initially assigned by the system to all new users. 'Others' covers all users outside the group.

A variant of the 'list directory contents' command (ls) is used to examine the access permission assigned to a file.

Typing ls –l followed by the pathname of a directory, or file, produces a 'long' listing which enumerates the characteristics of files, such as access permission, number of links (connections between disk blocks), name of the file creator, file size in bytes, date when the file was created or last changed, and the file name.

A typical dialogue which might occur when the ls –l command is issued is shown below:

```
% ls –l
-rw-rw----   1   john     97   Aug   17   10:51   Sdata
drwxr-xr-x   2   bob    1024   July   9   4:16    subd1
-rw-r-----   1   john    142   May   23   22:01   tg
```

Access permissions are specified by the first ten characters. The first character is:

d if the entry is a directory
– if the entry is a file

The next nine characters are grouped in three sets of three characters each. The first set refers to user (owner) permissions; the second set to group permissions; and the third set to permissions for all other users. Within each set, the three characters indicate permission to read, to write, and to execute the file as a command.

Character indication:

r read permission granted
w write permission granted
x execution permission granted
– indicated permission is not granted.

10.7. Default Protections.

The access permission is set in either absolute or symbolic terms. Absolute representations take the form of an octal number whose use is described in outline below, and in the chmod entry of section 1 of Volume I of the Programmer's Manual. The symbolic form takes the form of the access and user permission codes described in section 10.6, above.

File access permissions are initially assigned a default value that is controlled by the 'umask' associated with each user. To determine this setting, type

```
% umask
```

The system responds with a three digit octal value representing the binary complement of the default protections.

Suppose for example that the default file access permissions for a user are:

```
rwxr-x--x
```

which gives read, write and execute permission to the user's own newly created files, read and execute permission to users in the same group, and execute permission to all other users.

Arranging these permissions in groups of three gives:

```
rwx r-x --x
```

and representing them in binary form with 1 for a permission, and 0 for a refusal:

```
111 101 001
```

translates into the octal number

```
751
```

for the permissions granted. The octal complement of 751 is 26, or 000 010 110, which represents the permissions not granted and will be the value returned by the system when the umask command is used. For example:

```
% umask
026
```

To change the umask setting, the user types umask followed by the requisite octal value.

For example:

```
% umask 77
```

ensures that subsequently created files are accessible to no one but the owner.

For further details, see the creat, mknod and chmod references in section 2 of Volume I of the GENIX Programmer's Manual.

10.8. Changing Access Permissions.

The chmod command is used to modify read, write and execute permissions for files and directories, and uses the symbolic form of permission representation.

To change access permission, type chmod followed by the following arguments:

1. whose permission is to change: user (u), group (g), other (o), or all (a).
2. whether permission is to be: added (+), removed (–) or set (=)
3. type of permission being changed: read (r), write (w) or execute (x)

4. the name of the file or directory

For example:

To deny read and write access to the file called sdata, first obtain a listing of the current file permission settings, using the ls –l command:

```
% ls –l sdata
-rw-rw---- 1 John 97 Aug 17 10:51 sdata
```

In order to remove read and write access to users not the owner of the file, and outside the owner's group, the owner types:

```
% chmod o–rw sdata
```

The arguments to this command are the 'others' option, the 'remove read and write permission' sequence, and the filename. This example assumes that the user is within the directory containing the sdata file. If this were not the case, the full pathname of the file would be specified.

Chapter 11
Troubleshooting

11.1. Introduction.

The MG-1 Personal Workstation is a robustly designed and engineered system, manufactured using single-board techniques. The possibility of operational problems is therefore reduced to a minimum. As with any electronic equipment, care must be taken when physically handling its individual units, and when connecting or disconnecting the inter-unit leads.

The MG-1 does not incorporate user-servicable parts: in the event of the system becoming inoperable due to a disk crash or a unit being physically damaged, the supplier or support agency should be contacted for assistance.

This chapter describes the diagnostic facilities and software remedies available to the system administrator and user in the event of operational difficulties.

11.2. The System ROM.

Under normal operating conditions, the MG-1 System ROM automatically bootstraps the operating system from the hard disk.

Under faulty conditions the System ROM is intended to identify any faults which prevent the bootstrap procedure. In addition, the system ROM allows special-purpose diagnostic programs and stand-alone software to be loaded from floppy disk.

11.2.1. Power-on Tests.

The tests performed by the ROM upon power-up identify faults that would prevent more sophisticated diagnostic tools from being loaded.

The diagnostic LED is located under the front panel flap of the computer unit. Following power-on the system ROM illuminates the LED for one second to indicate the correct functioning of the processors, ROM and associated logic.

A sequence of tests is then performed by the System ROM which indicate detected faults by flashing a binary error code on the diagnostic LED. This acts as the error reporting system so as to prevent interruption of fault listing during display malfunctions.

When an error is detected the diagnostic LED repeatedly flashes a four-bit code (most significant bit first) using the following timings:

LED ON for 1 second:	bit = 1
LED ON for 0.25 seconds:	bit = 0
LED OFF for 0.25 seconds:	inter-bit gap
LED OFF for 2 seconds:	inter-word gap

These error codes pin-point failures in specific areas of memory and system logic. In the event of a system failure, report the nature of the failure and, when applicable, the error code sequence displayed by the diagnostic LED. The translations of these codes are given in Appendix E.

11.3. Forgotten Root Password.

To prevent unauthorised access to system files and maintenance routines, the MG-1 bootstraps into multi-user mode and requires the 'root' password to enter system maintenance mode.

This is in contrast to many UNIX type systems which power-up directly in system maintenance mode, giving any user direct access to sensitive system files.

The system administrator may opt to include a password procedure for the superuser or for all users. If the system administrator forgets the superuser password and a securely held record is not

available, entry to system maintenance mode can only be achieved by bootstrapping the MG-1 from the 'system security' floppy disk available from Whitechapel Computer Works.

Details of bootstrapping from a floppy disk are given in section 5.4.

11.4. Creating Additional File Space.

The complete GENIX operating system occupies some 14 Mbytes of storage, but because GENIX is not hierarchical, many files may be deleted or archived without compomising the system.

Many of the available files and utilities will be used infrequently, if at all, on some systems. The on-line manual for example occupies 1.4 Mbytes and can be regarded as dispensable if the hard copy manual is available. If this is the case, two options are available; total removal, and hence the permanent loss of the manual files, or the archiving of the files to floppy disk for subsequent restoration when required.

The rm command is used for the total removal of directories and files. For example, to remove the manual directory held in the usr directory, type:

```
% rm –rf /usr/man
```

There is an alternative to the full manual. In order to maintain the one-line description database "what is", run '/usr/lib/makewhatis' before removing the manual files.

Two further programs which serve no further purpose once the manual is removed are: the manual print program /usr/ucb/man (17 kbytes) and the manual formatter /etc/catman (13 kbytes).

Note that removal of a file requires write permission on its directory, but neither read nor write permission on the file itself.

For archiving to floppy disk, use the flar command. Ensure that a sufficient supply of formatted floppy disks are available by using the fdfmt command to format new or obsolete disks. The MG-1 formats floppy disks to a capacity of 800 kbytes, so two disks will be required to access the full manual and its support files.

To archive the on-line manual type:

```
% cd /
% flar c usr/man
```

A complete listing of files held on the system disk, together with their file size, is obtained with the following command:

```
% cd /; ls –Rcs
```

The games programs occupy 768 kbytes and can be removed with:

```
% rm –rf /usr/games
```

or archived with:

```
% cd /
% flar c usr/games
```

The text processing utilities also include candidates for removal on most systems. The typesetting utility /usr/bin/troff occupies 60 kbytes and is likely to be required by relatively few systems. Similarly the formatting program /usr/bin/nroff is unlikely to be used frequently. However, nroff should not be removed from systems maintaining the on-line manual since it is used by man for formatting. If both nroff and troff are to be deleted, or archived, the following macro packages can be removed:

/usr/lib/tmac/tmac.*	67 kbytes
/usr/lib/me	21 kbytes
/usr/lib/me/src	47 kbytes
/usr/lib/macros	75 kbytes
/usr/lib/term/tab*	30 kbytes
/usr/lib/deroff	22 kbytes

The following spell utilities (total 200 kbytes) may also be of limited use for some systems:

/usr/bin/spell	1 kbyte
spellin	11 kbytes
spellout	11 kbytes
/usr/lib/spell	17 kbytes
/usr/dict/hlista	56 kbytes
hlistb	56 kbytes
hstop	56 kbytes

The dictionary itself, /usr/dict/words, occupies 204 kbytes, but note that it is used by some of the games.

A further 204 kbytes of the system disk is occupied by text processing utilities 'diction', 'style' and 'explain':

/usr/bin/diction	1 kbyte
explain	1 kbyte
style	1 kbyte
/usr/lib/dict.d	9 kbytes
dprog	12 kbytes
explain.d	14 kbytes
style1	76 kbytes
style2	64 kbytes
style3	26 kbytes

The following mail programs may be considered superfluous to the needs of some systems (the = sign indicates a program link):

/bin/mail = /bin/rmail	25 kbytes
/usr/ucb/Mail = /usr/ucb/mail	68 kbytes

The Secretmail system comprises the following files:

/usr/bin/xsend	28 kbytes
/xget	28 kbytes
/enroll	23 kbytes

The Network Mail facility consists of five files:

/etc/delivermail	41 kbytes
/usr/lib/aliases	1 kbyte
/usr/ucb/newaliases	27 kbytes
/usr/ucb/prmail	14 kbytes
/usr/ucb/from	14 kbytes

MG-1 Systems which are not intended to be linked to other UNIX systems, either by local network or phone line, need not include the UUCP UNIX to UNIX copy routines (354 kbytes)

/usr/lib/uucp/*	183 kbytes
/usr/bin/uucp	30 kbytes
uudecode	17 kbytes
uuencode	11 kbytes
uulog	17 kbytes
uuname	13 kbytes
uupoll	19 kbytes
uusend	18 kbytes
uusnap	15 kbytes
uux	31 kbytes

The SCCS source code control system (591 kbytes) is only required for those MG-1 systems maintaining versions of user-written programs.

/usr/nsc/admin	64 kbytes
bdiff	29 kbytes
comb	48 kbytes
delta	72 kbytes
get	72 kbytes
sccs	36 kbytes
sccsdiff	2 kbytes
sccshelp	21 kbytes
prs	64 kbytes
prt	32 kbytes
rmchg	64 kbytes
unget	47 kbytes
val	23 kbytes
what	17 kbytes

Systems not utilising the RATFOR version of the FORTRAN programming language need not maintain the RATFOR translators (110 kbytes).

/usr/bin/struct	1 kbyte
/usr/lib/struct/*	89 kbytes
/usr/bin/ratfor	20 kbytes

If the C language program verifier, lint (298 kbytes), is not required, the following files can be removed:

/usr/bin/lint	1 kbyte
/usr/lib/lint/*	297 kbytes

MG-1 systems intended solely for single-user environments need not maintain the following files (123 kbytes):

/bin/who	14 kbytes
/bin/write	13 kbytes
/bin/wall=/etc/wall	13 kbytes
/etc/shutdown	17 kbytes
/usr/ucb/whoami	12 kbytes
/usr/ucb/f	22 kbytes
/usr/ucb/finger	22 kbytes
/usr/ucb/users	10 kbytes

Removing unnecessary entries from the terminal database /etc/termcap will release some 40 kbytes of hard disk space.

Systems utilising an editor other than those supplied can release 124 kbytes of hard disk space by the removal of the following linked files:

/usr/ucb/vi
view
e
edit
ex

Various files and directories expand during the day to day operation of GENIX.

The accounting file /usr/adm/wtmp is a record of who logs onto the system and their log in and log out times. It is possible for this file to expand to a significant size, and consideration should be given to its total removal or periodic truncation. For details of system accounting refer to the sa and ac commands in section 8 of Volume I of the GENIX Programmer's Manual.

In a multi-user system using the mail facility the file containing users' mail, /usr/spool/mail, can expand as mail continues to be deposited and left unread. This can also occur with the secretmail file /usr/spool/secretmail.

Systems using the uucp facility include files which can grow unnecessarily large, particularly in directories usr/spool/uucp and usr/spool/uucppublic. Refer to Volume 2 of the GENIX Programmer's Manual for details of administering UUCP related files.

The system uses two directories for temporary files /tmp and /usr/tmp which are cleared each time the system is started up. If the system remains running for long periods these directories can become full.

Systems which maintain the on-line manual include the set of directories /usr/man/cat*. The catman program formats the manual and stores the results in these directories. Alternatively, the directories fill up when manual entries are requested with the man command — but only if the cat directories exist. If therefore the ready formatted versions are not required remove the cat directories.

Many of these facilities, while not in use regularly can be useful over the long term life of the system. Accordingly, it is recommended that they be archived onto a separate storage system rather than being deleted completely.

11.5. Runaway Processes.

Very occasionally, either human or mechanical error causes a "runaway process" which resists interrupt or halt commands, and may produce an unwanted stream of output to the terminal. In the event of a runaway process carry out the following actions in sequence, until the process is halted:

1. Try pressing the <DELETE> key.
2. Try pressing the <Ctrl><\> sequence.
3. To completely reset the MG-1, type the keys in the four corners of the keyboard: <Esc><Alt><Scroll Lock><+>. This is an extreme measure and should only be used in the most severe circumstances. Damage to the file system is a distinct possibility when resetting the MG-1. To minimise this possibility, listen carefully to the hard disk drive (top centre of computer unit) and wait for ten seconds after a period of disk searching before switching off the mains power.

   Only reset the MG-1 as a last resort: it is not the correct way to shut down the system and can cause file inconsistencies which will need to be corrected with fsck.

11.6. The Diagnostic Floppy Disk.

The diagnostic floppy disk contains a range of GENIX commands for file and device handling, system checking, archiving, and maintenance. In the event of the hard disk being immobilised by a fault, the floppy disk can provide the troubleshooting facilities needed. Bootstrapping from a floppy disk is described in section 5.4.

Appendix A
Physical Specifications.

A.1. Processor

32-bit processor (NS 32016) with 8 MHz clock, floating point unit (NS 32081) and memory management unit (NS 32082) providing a full demand-paged virtual memory system with 1 kbyte page size.

A.2. Memory

Uses MOS semiconductor DRAM with 270nS cycle time. Dual ported between processor and display using 64 bits highway. Basic system 0.5 Mbytes, field upgradeable to a maximum of 8 Mbytes in 0.5 Mbyte or 2 Mbyte increments, all within the standard enclosure.

A.3. Display

Bit-mapped display system refreshed from system memory. Display controller uses a paged memory system compatible with the memory management unit, thus program variables can be used as screen buffers. Four screen maps are held concurrently in Video Mapping RAM with instantaneous switching permitting the use of double buffering techniques for smooth animation. Reverse video available.

A.4. Raster Graphics Processor

Hardware implemented RasterOp performs full set of logical operations between bit-aligned rectangles (2 dimensional arrays) with programmable rectangle size and stride length. The RasterOp processor operates on rectangles located anywhere in memory.

A.5. Input/Output

One serial (RS 232 C) port standard up to 9600 bps. General purpose expansion port allowing direct access to system bus and DMA service from the on-board controller. Accepts a mother board supporting upto 3 IBM PC compatible expansion boards.

A.6. Local Network

An integral IEEE 802.3 Ethernet controller is an option.

A.7. Fixed Disk System

5.25" Winchester technology fixed disk. Choice of capacities within the processor unit. Average seek time 50mS, transfer rate 600 kbytes/sec.

A.8. Floppy Disk System

5.25" double-sided, double-density, half-height floppy disk drive, 0.8 Mbyte formatted capacity.

A.9. Keyboard

Free-standing, lightweight solid state keyboard. IBM PC layout with 83 keys including numeric pad and function keys.

A.10. Pointing Device

A 3 button mouse is standard interfaced to the main system via a slave processor directly connected to the cursor.

A.11. Environmental Requirements

Noise output: 25dB Heat output: 150 watts Power requirements: 240 volts AC 13 amp 50Hz single phase 3 amp fuse.

A.12. Cabling

All connecting cables are supplied.

A.13. Physical Dimensions

Processor Unit: 490mm×165mm×465mm.
Screen (front): 440mm×385mm.
Screen (back): 205mm×210mm.
Keyboard: 450mm×190mm×30mm.

A.14. Weight

20Kg total.

Appendix B
Serial Port Pin Allocations.

The MG-1 supports both an inbuilt serial port and a Persyst ACC-2 two serial asynchronous port board running through the IBM bus. The pin allocations for the serial port are as follows:

Pin 2	transmit data	output
Pin 3	receive data	input
Pin 4	request to send	output
Pin 6	data set ready	input
Pin 7	signal ground	← →
Pin 8	carrier detect	input
Pin 20	data terminal ready	output

However, early models of the MG-1 have the following slight difference:

Pin 2	transmit data	output
Pin 3	receive data	input
Pin 4	request to send	output
Pin 5	carrier detect	input
Pin 6	data set ready	input
Pin 7	signal ground	← →
Pin 20	data terminal ready	output

For complete details of flow control and diagnostics for the serial ports, see the references to acc(4), tty(4) and newtty(4) in Volume I of the GENIX Programmer's Manual.

Appendix C
ASCII Codes.

A full list of the MG-1's ASCII codes can be found in file /usr/pub/ascii. The following table gives the octal values of the keyboard characters.

000 nul	024 dc4	050 (	074 <	120 P	144 d	170 x
001 soh	025 nak	051)	075 =	121 Q	145 e	171 y
002 stx	026 syn	052 *	076 >	122 R	146 f	172 z
003 etx	027 etb	053 +	077 ?	123 S	147 g	173 {
004 eot	030 can	054 ,	100 @	124 T	150 h	174 |
005 enq	031 em	055 –	101 A	125 U	151 i	175 }
006 ack	032 sub	056 .	102 B	126 V	152 j	176 ~
007 bel	033 esc	057 /	103 C	127 W	153 k	177 del
010 bs	034 fs	060 0	104 D	130 X	154 l	
011 ht	035 gs	061 1	105 E	131 Y	155 m	
012 nl	036 rs	062 2	106 F	132 Z	156 n	
013 vt	037 us	063 3	107 G	133 [	157 o	
014 np	040 sp	064 4	110 H	134 \	160 p	
015 cr	041 !	065 5	111 I	135]	161 q	
016 so	042 "	066 6	112 J	136 ^	162 r	
017 si	043 #	067 7	113 K	137 _	163 s	
020 dle	044 $	070 8	114 L	140 `	164 t	
021 dc1	045 %	071 9	115 M	141 a	165 u	
022 dc2	046 &	072 :	116 N	142 b	166 v	
023 dc3	047 '	073 ;	117 O	143 c	167 w	

Appendix D
Scan Codes

The following table gives the MG-1's keyboard scan codes.

Code	Unshift	Shift	Code	Unshift	Shift	Code	Unshift	Shift
-								
1	Esc		29	Ctrl		57	Space	
2	1	!	30	a	A	58	CLock	
3	2	@	31	s	S	59	F1	
4	3	#	32	d	D	60	F2	
5	4	$	33	f	F	61	F3	
6	5	%	34	g	G	62	F4	
7	6	^	35	h	H	63	F5	
8	7	&	36	j	J	64	F6	
9	8	*	37	k	K	65	F7	
10	9	(	38	l	L	66	F8	
11	0	)	39	;	:	67	F9	
12	-	_	40	'	"	68	F10	
13	=	+	41	#	~	69	NLock	
14	Del		42	Shift		70	SLock	
15	Tab F	Tab B	43	\	|	71	Home	7
16	q	Q	44	z	Z	72	Up	8
17	w	W	45	x	X	73	Pg Up	9
18	e	E	46	c	C	74	-	
19	r	R	47	v	V	75	Left	4
20	t	T	48	b	B	76	5	
21	y	Y	49	n	N	77	Right	6
22	u	U	50	m	M	78	+	
23	i	I	51	,	<	79	End	1
24	o	O	52	.	>	80	Down	2
25	p	P	53	/	?	81	Pg Dn	3
26	[	{	54	Shift		82	Ins	0
27	]	}	55	*	Prt Sc	83	Del	.
28	CR	CR	56	Alt				

Appendix E
Error Codes

The tests provided by the MG-1 System ROM upon power-up are intended to be as simple as possible; they will identify only those faults that would prevent more sophisticated testing tools from functioning properly.

Before commencing tests, the ROM will illuminate the indicator for one second, then switch it off. This shows that the processor unit, ROM and associated tty are working.

Reading the error codes has been explained in section 11.2 of this Guide. The codes translate as follows:

E.1. Miscellaneous

0 **ROM decode error**
1 **SRAM error**
2 **IOP not initialised**
3 **Video Map error**
4 **ICU error**
5 **USART error**
6 **Processor trap**
7 **undefined**

E.2. DRAM Tests

8,9,A,B **Single-bit error**
C **Multi-bit error**
D **unused**
E **Addressing error**
F **Refresh error**

Appendix F
Reading List.

F.1. General.

A wide range of UNIX text books and introductory guides already exists. These are widely available and are updated and expanded continually. Similarly, there are a large number of good general computer science references on such topics as interactive graphics, networking and compilers. There would be little point in listing these in detail, especially as most such references include their own bibliographies. Whitechapel Computer Works is currently engaged in extending its range of MG-1 documentation. Additions will be announced.

F.2. GENIX.

The major reference is

"GENIX Programmer's Manual", 2 Vols., (1985), NSC.

This manual is the essential source of information on the components and use of the GENIX operating system. Volume I covers the GENIX commands and application programs, system calls, subroutines, special files, file formats, games, and system maintenance routines; Volume II presents a series of papers explaining the use of these facilities, and is especially useful in its coverage of the utilities.

F.3. The C Programming Language.

The basic reference on C, also known as the "White Book", is

"The C Programming Language" by Kernighan,B.W. & Ritchie, D.M., (1978), Prentice Hall, Inc.

Also recommended are

"The C Primer" by Hancock,L. & Krieger,M., (1982), McGraw Hill.

"C Programming Guide" by Purdam,J., (1983), Que Corporation.

"Learning to Program in C" by Plum,T. (1983), Plum Hall, Inc.

"The UNIX Programmer's Manual" by Thompson,K.L. & Ritchie,D.M., (1978), Bell Laboratories.

F.4. The General Programming Environment.

The major MG-1 programming languages such as C, FORTRAN, and Pascal are all amply covered in general texts or in their own compiler documentation. Assembler routines may be incorporated within program modules written in any of the above languages. The most useful references for the GENIX Assembler are the GENIX Programmer's Manual, and the following NSC publications:

"NS32000 Instruction Set Reference Manual"

"Series 32000 Cross Assembler Reference Manual"

F.5. Hardware.

Many of the relevent references are published by the National Semiconductor Corporation. Each of the processors introduced in Chapter 3 of this manual is covered in its own technical manual, which together make up a group of documents called "The 3200 Microprocessor Family". Also available through NSC is a series of documents such as "The Benefits of Demand Paged Virtual Memory", and "Introduction to the NS32000 Architecture".

Index