CONTENTS

CONTENTS

Figure

Table

INTRODUCTION AND OVERVIEW

GENERAL DESCRIPTION

Millennium System's 9520 Software Development System (figure 1-1) is a general purpose, low-cost, user-oriented Z80A microcomputer system that is designed to meet the needs of the programmer in a program development environment. The user can furnish his own display terminal console or a display terminal console can be supplied with the system, at the users option. The 9520 is a self-contained (less display terminal console) system that provides all of the capabilities for the development of user programs. Assemblers, compilers, editors, and document creation packages are available. The 9520 is a Z80A based system with 64K memory standard (and additional 48K of memory is an available option) and two integral eight-inch, double density floppy disk drives. Interface connections are three RS-232-C ports, one RS-422 port and an IEEE-488 port. These interfaces allow the 9520 to connect with a display terminal console, a printer, a hardware emulator (such as the companion 9508), as well as any other compatible peripherals.

HARDWARE FEATURES AND CAPABILITIES

Processor Board

The single printed circuit processor board (figure 1-2) in the software development system contains all the circuits necessary to perform the data manipulations required to develop software. In addition, the printed circuit board has line drivers and receivers for two-way communication with the display terminal, and to upload and download software to a debugging system. The processor board also has a self-test feature, which tests the on-board memory array, baud-rate and interrupts, the two asynchronous receiver/transmitters, the DMA capability, the floppy disk controller and the IEEE-488 port.

Power Supply

The power supply furnishes the +5 VDC and $\pm$12 VDC required by the processor board plus the +5 VDC and +24 VDC used by the disk drives.

Disk Drives

The disk drives for the 9520 Software Development System allows data to be formatted in single or double density on standard 8-inch floppy disks. The reading or writing of single or double density (FM or MFM respectively) is controlled by the user (as selected at format time).

Display Terminal

The display terminal may be user furnished or a Millennium 9501 Display Terminal Console be supplied with the 9520 Software Development System. The only prerequisite for a user supplied display terminal is it must be able to display 80 character lines and be compatible with the RS-232C interface.

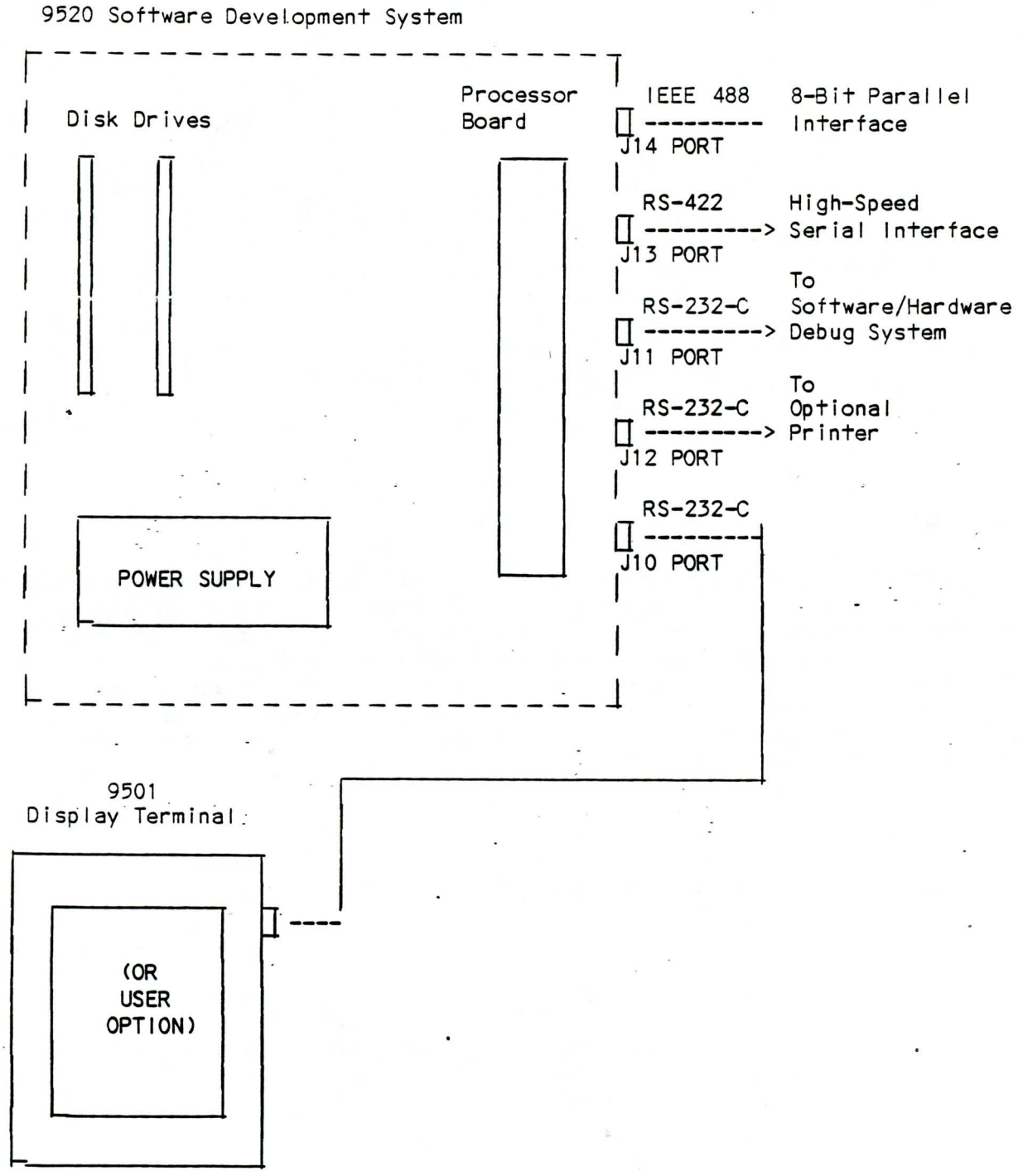

9520 Software Development System
Disk Drives
Processor Board
IEEE 488 8-Bit Parallel Interface
J14 PORT
RS-422 High-Speed Serial Interface
J13 PORT
To Software/Hardware Debug System
RS-232-C
J11 PORT
To Optional Printer
RS-232-C
J12 PORT
RS-232-C
J10 PORT
POWER SUPPLY
9501 Display Terminal
(OR USER OPTION)

Figure 1-2.

SOFTWARE FEATURES

This section presents a description of the 9520 software features. It provides an overview of the disk operating facilities.

Overview of Software Resources

Two operating systems may be used with the 9520, MP/M™ or CP/M™. MP/M is a priority-driven, disk operating system that provides the user with a multi-tasking software environment. CP/M is a single-user operating system. These operating systems provide the following capabilities:

- Creating and editing user source program files using the Edit utility.
- Storage and management of files in source and object format.
- Assembling and linking of object files for conversion into executable load modules.
- Checkout and dynamic debug of source programs originating in the MP/M and CP/M environment.
- Upload and download intersystem communication for the transfer of data between the 9520 Development System and a remote hardware/software in-circuit-emulator station.

User Interface

The user communicates with the software development system by entering input data and commands at the display terminal keyboard. The keyboard commands are entered as a single-line character string. These commands access the edit utility, file utilities, I/O utilities, assembler/cross-assembler and other software routines. The system responds by displaying data and messages at the terminal which prompt the user for additional input, as necessary, to enable processing to be completed.

The user programs originated at the 9520 Software Development System can be downloaded to a remote in-circuit emulator station (e.g. Millennium's 9508 or 9516). The users program can then be executed to locate and correct programming errors and/or detect and correct malfunctions in the microprocessor system circuits that are subjected to hardware/software integration tests.

Software Packaging

The system programs reside on three standard floppy diskettes. Two of the diskettes contain the operating system program along with one or more additional sets of programs as indicated below. The remaining diskette contains the system diagnostics.

™ MP/M, CP/M are Trade Marks of Digital Research of Pacific Grove, CA 93950

Diskette #1 (SYSTEM).. Operating System and Utilities
#2 (LANGUAGE TRANSLATOR).. Cross-Assembler, Linker
#3 (DIAGNOSTIC).. System Diagnostic Monitor and Test Routines

Separate diskettes are available for optional high-level programming languages (such as C Compiler and PASCAL Compiler). These compilers are used to generate object code output for 8080, 8085 and Z80 microprocessor-based systems.

The Wordstar™ Text Editor utility is a standard software feature provided with the operating system diskette.

The operating details for programming elements of the MP/M (Multi-Programming Monitor Control Program) and CP/M (Single-Console Operating System) are provided in separate Users Guide manuals that are shipped with the 9520 Software Development System.

Information in this manual describes how the MP/M, CP/M Operating System is used. With this knowledge, the user learns how to invoke the Millennium System Utility modules to originate and assemble user programs on the 9520 Software Development System and build load modules that can be downloaded and executed on a remote software/hardware debugging station.

Operating System Environment

The development system software consists of the following sets of programs:

- o Operating System
- o System Utilities
- o Language Translator
- o System Diagnostics

The operating system is based on a memory resident Executive that provides a variety of services to the system and user programs. The Executive provides all of the interaction between the hardware and software, and between the system and the user. All queries to the input/output devices are processed through the Executive. The Executive includes interrupt handling, I/O processing, supervisor call processing, user communications and disk file management routines. The MP/M operating system is easily configured with optional user memory segment sizes, resident process modules, relocatable modules, and quantity of terminals. The description of the system generation procedure for MP/M is given in chapter 7. The CP/M operating system requires no system generation.

™ Trade Mark of MicroPro International Corporation, San Rafael, CA 94901

System Utilities

The utility programs provide various file management functions such as editing, renaming, deleting, copying and displaying existing files. The utilities also include programs which allow the user to access and change certain system parameters (e.g. baud rate, BRATE utility) and submit a series of commands from a file for batch processing, rather than entry from the keyboard (e.g. SUBMIT utility). The FDISK utility allows a user to format the diskette for single or double density applications, duplicate a complete diskette and copy systems tracks from one diskette to another.

Except for the Text Editor (EDIT) and Dynamic Debugging Tool (DDT) utilities, the description of utility programs are described in chapter 5. The text editor and debugging utilities are described in chapter 6.

Language Translator

The language translator utility is a relocatable macro assembler which translates the target microprocessor assembly language programs into object code which is executable in the users remote microprocessor system. The following microprocessors are supported via the cross-assemblers:

- o Z80A
- o 8086/88
- o 8080/85A
- o 8048/49/41/21/35/39
- o 6800/01/02/03/09

Each cross-assembler program is described in a separate publication supplement that is listed in the preface. The diskette containing the cross-assembler utility also includes the linker and formatter/downloader software. The linker is described in chapter 5.

The language translator utility operates to accept the source program files that are written in Z80A assembly language with the Text Editor and converts these source files into *object files that can be input to the linker program.

*NOTE: Except for the 8048 family cross-assemblers, all cross-assemblers produce relocatable code.

The linker utility links relocatable object files into executable load modules by resolving all external references to the file while building the load module.

System Diagnostics

The 9520 System Diagnostic program provides the user with a comprehensive set of programs to diagnose and test system components. It is designed so that the user can determine whether a malfunction is caused by a program error or a fault in the hardware.

The System Diagnostic programs are contained on a separate diskette. The diagnostic monitor and test routines are described in a separate publication's supplement that is listed in the preface.

SPECIFICATIONS

GENERAL

Millennium System's 9520 Software Development System provides a compact software development tool housed in a single module consisting of dual-floppy disk drives, a single power supply and a Z80 based, single board CPU with integral I/O controller and memory resources.

9520 PHYSICAL CHARACTERISTICS

Dimensions

Height - 13 5/16 inches (33.3325 Cm)

Width - 19 3/16 inches (48.4475 Cm)

Depth - 24 1/8 inches (61.085 Cm)

Weight - 74 lbs (33.6 kilogms)

Environmental Limits	Operating	Storage
Ambient Temperatures =	40° to 104°F (4.4° to 40°C)	-8°F to 117°F
Relative Humidity =	20% to 80%	1% to 95%
Maximum Wet Bulb =	78°F (25°C)	No Condensation

Cooling

System cooling is provided by two, AC operated fans, mounted at the rear of the enclosure, that draws cool air through the disk drives, over the power supply and accross the single printed circuit board.

***** CAUTION *****

The system top cover must be replaced before AC power is applied. Operation without the top cover will defeat the cooling function of the fans and damage to the disk drives, power supply and/or the printed circuit board may occur.

AC Power Requirements

50/60 Hz ± 0.5 Hz
100/115 ± 10% VAC
200/230 ± 10% VAC

Fuses for Rated Voltage

Voltage	Power Supply and Fan (F1)	Disk Drives (F2)
100/115 VAC	3AG, 5 Amp	3AG, 1.0 Amp
200/230	3AG, 2.5 Amp	3AG, 0.75 Amp

Front Panel Controls

POWER PUSH ON/OFF — The power on/off switch is illuminated (red) when AC power is applied to the Software Development System. When the push-switch is illuminated, this indicates not only the presence of AC voltage but also the presence of +5 VDC.

DIAG INT — The DIAG INT (diagnostic interrupt) push-switch is used when the diagnostic disk is inserted in the disk drive. The switch must not be used when the Software Development System is operating.

RESET — When activated, the RESET push switch reinitializes the internal microprocessor and forces the system to jump to the on-board self-test program.

Input/Output Rear Panel Ports and Switches

J10-CONSOLE — Standard RS-232C, D-type, 25-pin (DCE configuration) female connector. Connector J10 is generally connected to the Display Terminal.

J11-RS-232-1 — Standard RS-232C, D-type, 25-pin (DCE or DTE configuration option) female connector. Connector J11 is generally connected to the software/hardware debug system.

J12-RS-232-2 — Standard RS-232C, D-type, 25-pin (DCE configuration) female connector. Connector J12 is generally connected to the printer.

J13-RS-422 — Standard RS-422A, D-type, 37-pin female connector. Connector J13 is generally connected to a unit with serial, high-speed input/output transfer requirements. (Corresponds to RS-449 configuration.)

J14-IEEE 488 — This connector is a standard, 24-pin, female, high-speed parallel transfer I/O port. The port provides eight-data/address lines, eight control lines and eight signal and frame ground lines.

J15 POWER ON — Standard AC, male, connector for power input.

S5 BAUD RATE — S5 is an eight-position rotary switch used to select the desired baud rate. The switch position is read by the system and can be changed each time the system is reset.

Positions:	
0	110
1	300
2	600
3	1200
4	2400
5	4800
6	9600
7	19200
8	----
9	----

S6 DEVICE ADDRESS — The DEVICE ADDRESS switch is a five-position DIP switch that controls addressing of the peripherals connected to the IEEE-488 (GPIB interface) port.

Power Supply Specifications

The 9520 power supply provides the four DC operating voltages necessary for the operation of the single processor board and the disk drives. The specifications for the power supply are as follows:

AC Input

90 to 130 volts or 180 to 260 volts AC
47 to 63 Hz, single-phase
(Power Supply input is adjusted at Millennium in accordance with user requirements.)

DC Outputs

Voltage	Current
+5 volts	20A max.
+12 volts	5A max.
-12 volts	3A max.
+24 volts	3.5A max.

Maximum Output Power 150 Watts

Regulation Characteristics

Line: ±0.1% for a full line change of 90-130 VAC or 180-260 VAC.
Load: ±0.1% for a 100% load change.

Hold-up Time

20 milliseconds minimum after AC input voltage is lost.

Output Noise

50 mv PK-PK maximum

Disk Drive Specifications

The specifications for the disk drives are as follows:

Performance Specifications

Capacity	Single Density	Double Density
Unformatted		
Per Disk	3.2 megabits	6.4 megabits
Per Track	41.7 kilobits	83.4 kilobits
IBM Format		
Per Disk	2.0 megabits	n/a
Per Track	26.6 kilobits	n/a
Transfer Rate	250 kilobits/sec	500 kilobits/sec
Latency (average)	83 ms	8 ms
Access Time		
Track to Track	8 ms	8 ms
Average	260 ms	260 ms
Settling Time	8 ms	8 ms
Head Load Time	35 ms	35 ms

Functional Specifications

	Single Density	Double Density
Rotational Speed	360 rpm	360 rpm
Recording Density (inside track)	3200 bpi	6400 bpi
Flux Density	6400 fci	6400 fcik
Track Density	48 tpi	48 tpi
Tracks	77	77
Physical Sectors		
SA800	0	0
SA801	32/16/8	32/16/8
Index	1	1
Encoding Method	FM	MFM/M^{2}FM
Media Requirements		
SA800	SA100/IBM Diskette	SA102/IBM Diskette
SA801	SA101	SA103

9501 Display Terminal Console

The 9500 Family Software Development System can be purchased with an optional display terminal. The following paragraphs list the physical characteristics of the 9501 Display Terminal Console.

Display Cabinet Dimensions

Height - 14.0 inches (35.6 Cm)
Width - 16 1/2 inches (41.9 Cm)
Depth - 14 1/4 inches (36.2 Cm)

Keyboard Dimensions

Height - 3.0 inches (7.6 Cm)
Width - 16 1/2 inches (41.9 Cm)
Depth - 7 1/2 inches (19.0 Cm)

Cabinet Weight - 30 lbs (13.6 kilogms)
Keyboard Weight - 4.5 lbs (2.3 kilogms)

Environmental Limits	Operating	Storage
Ambient Temperature	32°F to 122°F (0°C to 50°C)	-40°F to 149°F (-40°C to 65°C)
Relative Humidity	10% to 95% (Non Condensing)	N/A (No Restriction)

AC Power Requirements

50/60 Hz ± 0.5 Hz
100/115 ± 10% VAC
200/230 ± 10% VAC

Fuses for Rated Voltage

VOLTAGE	FUSE
100/115 VAC	3AG, 1 Amp
200/230 VAC	3AG, 0.5 Amp SLO-BLO

9501 Display Terminal Controls

Front Panel Controls

Other than the movable keyboard, as shown in figure 2-1, there are no front panel controls. The keyboard interfaces with the display terminal via an expansion cable (similar to a telephone cable).

Rear Panel Controls and Connectors (figure 2-2)

The controls on the 9501 Display Terminal rear panel are the POWER switch, CONTRAST control, the FUSE holder, the BAUD RATE setup switch and the FUNCTION setup switch.

POWER Switch

The POWER switch is a rocker-type switch. When the end of the switch with the white dot is pressed, AC power is applied to the 9501 Display Terminal. Pressing the unmarked end of the rocker switch removes AC power. One second after the white-dotted end of the POWER Switch is depressed, an internal beeper will beep indicating the presence of AC power.

CONTRAST Switch

The CONTRAST switch is a rotary switch that controls the white to black intensity of the visible display. The switch can be set to the users satisfaction.

S1 BAUD RATE (SIP) Switch

The S1 BAUD RATE select switch is a 10-position, bit selection switch. The bit selection positions determine the baud rate input into the display terminal, the baud rate output from the display terminal, number of stop bits and word length. The following illustration (figure 2-3) shows the bit allocations and the baud rate select bit configurations.

S2 FUNCTION Select Switch (SIP)

The S2 FUNCTION select switch is a 10-position, bit selection switch. The bit selection positions select operational functions for the display terminal. The following illustration (figure 2-4) shows the bit allocations, the I/O mode and the parity selection.

SEL SW 115/230 VAC SWITCH BLOCKING STRIP — The position of this left/right switch must coincide with the primary AC source voltage. The left position of the switch indicates the 115 VAC source; the right position of the switch indicates the 230 VAC source. The SWITCH BLOCKING STRIP prevents the inadvertent changing of the switch.

FUSE 115 VIA/ 230 0.5 A — The FUSE holder contains the fuse that protects the internal circuits.

****

* *

* - CAUTION - *

* *

* The intent of the fuse should never be *

* ignored, nor should any attempt be made *

* to defeat the intention of this protec- *

* tive device, as damage to the equipment *

* may occur. *

* *

****

P3 (RS-232) This is a 25-pin, female, RS-232C connector port that is generally connected to the host source. The internal configuration of the connector is shown in table 2-1.

Table 2-1. Host Interface Connector Configuration

PIN No.	SIGNAL NAME
1	Frame Ground
2	Transmit Data Output
3	Receive Data Input
4	Request To Send Output
5	Clear To Send Input
6	Data Set Ready Input (opt.)
7	Signal Ground
8	Carrier Detect Input
20	Data Terminal Ready Output
9	20 mA source (+12V, no load)
14	20 mA source (+12V, no load)
10	Detected current loop data
25	Current Loop +, Transmit*
13	Current Loop -, Transmit*
12	Current Loop +, Receive*
24	Current Loop -, Receive*

P3 (PRINTER) This is a 25-pin, female, RS-232C connector port that may be connected to an optional printer. The internal configuration of the connector is shown in table 2-2.

Table 2-2. Printer Connector Configuration

PIN No.	SIGNAL NAME
1	Protect Ground
2	Transmit Data
3	Receive Data
4	Request To Send
5	Clear To Send
6	Data Set Ready
7	Signal Ground
8	Data Carrier Connect
20	Data Terminal Ready

Power Cord

The power cord applies the AC power to the 9501 Display Terminal. One end of the power cord is permanently affixed to the 9501. The free end is a standard 3-prong connector. If the equipment is used for international applications, remove the U.S.-style connector from the power cord and install a connector that will mate with the local power receptacle. Power cord wires are color-coded as follows:

green = earth ground,
black = primary power (hot),
white = primary power return (neutral).

P6

The P6 receptacle accepts the expansion cord from the keyboard. The receptacle is configured as a snap-type modular receptacle.

INSTALLATION AND CHECKOUT

GENERAL

This chapter describes the procedures for installing the 9520 Development System components at the users site. The user may install his own display terminal or the optional 9501 Display Terminal that is available for use with the equipment. Installation involves the inspection and set up of components, connecting the Display Terminal Console to the 9520 Development System and conducting the system power on check to verify the operational readiness of equipment.

UNPACKING AND VISUAL INSPECTION

All of the hardware and software items that are required to install and operate the equipment at the users site is shipped in packaged units. External cables are included for connecting the display terminal to the 9520 Development System. All items shipped for the standard system configuration are described in table 3-1.

The 9520 Development System (and optional 9501 Display Terminal Console, if provided) was thoroughly inspected and checked out at the factory prior to packaging for shipment to a customer. After removing the equipment from it's box, inspect for scratches, dents or other damage that might have occurred during shipping. Refer to the shipping papers to verify that all the components are present.

If physical damage is evident when received, do not operate the equipment. File a claim with the shipping firm immediately and notify Millennium System's Customer Service department at once. Millennium will arrange for repair or replacement of the equipment without waiting for settlement of the claim against the carrier.

If the equipment must be returned to Millennium, attach a tag showing the owner, address, serial number, and a description of the failure. The original shipping carton and packing material should be reused with the RMA (Returned Material Authorization) number prominently displayed. An RMA number can be obtained by calling Customer Service on the toll-free, hot-line numbers listed in the preface.

Millennium System's Technical Support Representatives and Customer Engineers are available to provide consultation and assistance on request.

Table 3-1. Packaged Items for Standard 9520 System Configuration

QTY	ITEM DESCRIPTION
1	9520 Development System Unit
1	AC Power Cord
1	RS-232 Interface Signal Cable (for Display Terminal)
1 (User Option)	9501 Display Terminal Keyboard Unit (TeleVideo Model #950)
3	Standard Diskettes which contain the following software: a) Operating System Software b) Language Translator Software c) System Diagnostic Software
1 Pkg	Document Package which consists of the following manuals: a) MP/M™, CP/M™ User Guide b) WordStar™ Text Editor User Guide c) 9520 Development System Users Manual d) 9501 Display Terminal Operators Manual (provided with optional Display Terminal)

MAINTENANCE AND SERVICE POLICIES

Unless notified to the contrary, any claims for operations assistance and/or service will be provided by Millennium Systems, Inc., from its plant in Cupertino, California. Should assistance be required, call Customer Service.

AC INPUT REQUIREMENTS

The 9520 Development System is wired by the manufacturer to accept AC power input for 100V/115V and 200V/230V at 50 or 60 Hz. Before installing the equipment, check the power specification label on the back panel to ensure that AC input requirements for the equipment coincides with the facility supply level. If a discrepancy is noted, do not attempt to make adjustments and contact the Millennium Customer Service Representative.

INTERFACE SWITCH SETTINGS

The development system employs three hardware interface switches, SW1, S5 and S6 that adapt the system for the users operating requirements.

Two of the switches, S5 (Baud Rate) and S6 (Device Address), are located at the rear of the development system chassis. The settings for these switches is determined by the system operating characteristics as described in subsequent paragraphs.

The Disk Alignment Switch (SW-1) is located on the processor printed circuit board inside the Development System chassis. This switch adapts the system for compatibility with the type of display terminal interfaced by the user and enable disk alignment aid diagnostics.

Baud Rate Switch

The Baud Rate Switch (S5) is an eight-position, rotary switch. Each position of the switch selects a specified baud rate within the range of 110 through 19,200 as indicated by the label and associated calibration marks for each switch position.

All other baud rates can be selected by the software after the system is initialized. The BRATE utility program permits a user to examine and modify the baud rate or status line usage for the assigned port. The BRATE utility is described in chapter 5.

Device Address Switch

The Device Address switch (S6) is a five-position DIP switch that is used to assign the hexadecimal, I/O device, address for the IEEE-488 parallel port at connector J14. The user can position the bit switches to select one out of a possible 32 combinations ($_{25}$) for the address assignment.

Disk Diagnostic Switch

The Disk Diagnostic Switch (SW1) is a four-position DIP switch that is used to select the type of display terminal interfaced with the system and to enable the Disk Alignment Diagnostic program. The switch positions SW1-1 through SW1-4 must be set as follows to interface the display terminal to the 9520:

NOTE: The positioning of SW1 to enable the Disk Alignment Diagnostic is not required for installation, but is described in the 9520 System Diagnostic Manual.

SW1-1	SW1-2	SW1-3	SW1-4	FUNCTION
OFF	OFF	OFF	OFF	BOOT, TELEVIDEO 950 TERMINAL
OFF	OFF	OFF	ON	BOOT, ANY OTHER TERMINAL TYPE
OFF	X	OFF	X	BOOT, CHECK SYSTEM IDENTIFICATION BYTES SINGLE DENSITY
OFF	X	ON	X	BOOT, NO CHECK SYSTEM IDENTIFICATION BYTES SINGLE DENSITY
OFF	ON	OFF	OFF	
OFF	OFF	X	X	PRESENTLY NOT USED
OFF	OFF	ON	OFF	
ON	X	X	X	ENABLE DISK ALIGNMENT AID PROGRAM
ON	OFF	OFF	OFF	SEEK TRACK 00
ON	OFF	OFF	ON	ALTERNATELY SEEK TRACKS 00 AND 01
ON	OFF	ON	OFF	READ TRACK 1
ON	OFF	ON	ON	READ TRACK 37
ON	ON	OFF	OFF	READ TRACK 38
ON	ON	OFF	ON	READ TRACK 39
ON	ON	ON	OFF	READ TRACK 76
ON	ON	ON	ON	WRITE TRACK 76

PREPARATION OF EQUIPMENT FOR USE

Use the following procedures to set up the equipment and complete the external cable connections.

9520 Development System Preparation and Set Up

1. Open the disk drive access doors and remove the cardboard Shipping Disk that is installed for packing purposes. The shipping disk is installed by the manufacturer to avoid possible damage to read/write heads when the equipment is either transported or stored for long periods. This shipping disk must be inserted by the user (with the heads positioned at track 76) whenever the equipment is packaged for reshipment.

2. Verify that the Baud Rate and Device Address switches (S5 and S6) located on the rear of the chassis, are adjusted as described in previous paragraphs. If the IEEE-488 I/O port at connector J14 is not used, it is not necessary to adjust the Device Address Switch (S6).

9501 Display Terminal Preparation and Set Up

The preparation and set up requirements for the 9501 Display Terminal are described in the Operators Manual that is shipped with the equipment. Refer to the manual for the various switch settings, configuring of interface connectors and setting the desired baud rates, work lengths and stop bits.

The 9501 Display Terminal Console is equipped with an internal self-test diagnostic, which can be conducted in a stand-alone mode, i.e., without the need for connecting the terminal to other equipment.

The RS-232 cable that is shipped with the 9520 Development System is used for the test connections as described in the procedure which follows. The self test diagnostic performs a check of the video attribute functions and the communications path between the printer output port and terminal input port. The self test is to be conducted as follows:

1. Connect the power cable to the facility AC source.

2. Position the four Terminal Baud Rate switch positions (on S1 Baud Rate) 7, 8, 9, 10 to be identical to the Print Baud Rate at switch positions 1, 2, 3, 4, respectively. This sets the Baud Rate for the printer output port to the same rate as the RS-232 input port.

3. Switch positions 5 and 6 (of S1) are not to be changed for the Self-Test.

4. Connect the RS-232 Cable from the connector labeled P3 to P4.

5. Turn on the Display Terminal power switch. Allow 10 to 15 seconds for the screen to warm up and display the cursor in the upper left-hand corner of screen.

6. Press and hold the SHIFT key while pressing the SET-UP/NO-SCROLL key.

7. Press and release the 1 key.

   The screen should display all available characters and attributes for reverse video, grey shade, blinking line and underscore.

8. Press and release the Z key.

   a) All but the bottom line of the screen should be cleared.

   b) After approximately 3 seconds, the word PASS should be displayed in the upper left-hand corner of the screen to indicate the test was successful.

   c) If FAIL 2 appears, verify that Baud Rate switches are set the same for both ports. Also verify that the RS-232 cable is securely connected between P3 and P4 and repeat the test.

   d) If the problem still persists, refer unit/problem to Millennium Customer Service personnel.

External Cable Connections

External cable connections for interfacing the users peripherals to the 9520 Development System are given in the following table.

FROM: CONNECTOR ON 9520 CHASSIS	I/O PORT DESCRIPTION	TO: USERS EXTERNAL PERIPHERAL
J10	RS-232-C Interface	Display Terminal Unit
J11	RS-232-C Interface	Software/Hardware Debug System Unit

(continued)

J12	RS-232-C Interface	Users Optional Printer Unit
J13	RS-422 Interface	Users Optional High-Speed Serial I/O Interface Device
J14	IEEE-488	Users Optional Parallel Interface I/O Device

POWER ON CHECK

CAUTION: The diskette should not be engaged with the disk drive during any initial power-up sequence as damage to the diskette may result.

The power on check is performed by the system to verify the operational readiness of hardware components whenever the system is powered up or restarted. The power on check is implemented by a self-test, bootstrap program that is permanently stored in the 2716 EPROM firmware on the printed circuit board. The program diskette should always be removed from the disk drives (or if the diskette is installed, the access door should remain open) whenever AC power is applied to, or removed from the system.

The 9520 Boot PROM diagnostic is automatically initiated when the POWER switch is turned ON to power up the system from a cold start, or when the RESET switch is operated to reset system operations. The Self Test diagnostic performs an operational check of on-board circuit functions. The status of the test is presented to the operator by means of eight LED indicators that are located on the printed circuit board and a message that is displayed on the terminal screen.

Initial Start Up of System

Use the following procedure to power up the system from a cold start:

1. Turn on the POWER switch at the display terminal. The physical location of this switch will vary with different terminals, and may be located at the front, rear or side of the terminal chassis.

   The terminal response when power is applied will also vary with different terminals. A typical response is as follows:

   a. The terminal bell will beep within 1 second to indicate power is on.

   b. After 10 to 15 seconds, the cursor will appear in the upper left-hand corner of the screen.

At this time, the operator can adjust the contrast (or intensity) control to obtain the desired brilliance for the screen.

2. Verify the diskette is not mounted on the disk drives and turn on the POWER switch at the 9520 development system control panel.

3. Observe the terminal display which presents the following message line to indicate the Boot PROM diagnostic is running.

```
9520 SELF TEST VERS 2.0 CO
```

The diagnostic routine runs for approximately three seconds. A total of eight circuit functions are checked by the program. Each circuit function is coupled with an LED which illuminates to provide status to the operator. The displayed message line spells out the word C-O-M-P-L-E-T-E as each circuit function passes its test. The LEDs are numbered 1 through 8, beginning at the rear of the chassis and counting forward.

NOTE: The LEDs are located under the top cover of the 9520 chassis on the top edge of the printed circuit board. It is not necessary to remove the top cover unless a malfunction is detected by the diagnostic (see step 5).

The following circuit functions are tested in sequence with the corresponding LEDs and display message providing status for the test results:

DISPLAY MESSAGE	TEST SEQUENCE LED #	ON-BOARD CIRCUIT TESTED
	1	Baud Rate Generator and Interrupt Controller (Z80-CTC Counter/Timer)
CO	2	Serial RS-232 I/O Port Dual Asynchronous Receiver/ Transmitter (DART 2)
COM	3	Serial RS-232 I/O Port Dual Asynchronous Receiver/ Transmitter (DART 1)
COMP	4	Floppy Disk Controller
COMPL	5	Parallel IEEE-488 I/O Port Controller
COMPLE	6	RAM Memory Locations 8000H - FFFFH
COMPLET	7	RAM Memory Locations 0000H - 7FFFH
COMPLETE	8	DMA Controller

All of the LEDs are turned on initially when the system is powered up or reset. As the diagnostic steps through each circuit under test, the validity of each circuit function is verified. The LED is turned off, and one of the message characters is displayed, if the circuit passes the test. The diagnostic then advances to test the next circuit in sequence. This testing sequence is repeated for each of the tests. the program will then read the state of the SW1 DIP switch on the printed circuit board to verify the switch is off. The off position indicates the Disk Alignment Aid routine is not selected and the operating system program can be booted into memory from the system diskette.

4. If no malfunction of the hardware is detected by the diagnostic, the following message is displayed on the terminal screen to notify the operator that the system diskette can be installed and loaded into memory to generate the system parameters and begin system operations as described in chapters 7 and 8.

9520 SELF TEST VERS 2.0 COMPLETE

Resolving Start Up Problems

If a malfunction is detected by the self test diagnostic, the LED associated with the failed circuit remains illuminated, the displayed message line is not completed, and the program will halt on the error. One of the following error messages is displayed at the terminal screen to notify the operator that a malfunction is present at startup:

ERROR MESSAGE	LED #	DISPLAY CHARACTER
CKSUM ERROR NO STACK ERROR TIMER ERROR	1	
RDR/PUN ERROR CONSOLE ERROR	2	C0
RS422 ERROR REM10 ERROR	3	M
INVALID INT ERROR	4	P
	5	L
RAM 8XXXH ERROR RAM CXXXH ERROR	6	E
RAM 0XXXH ERROR RAM 4XXXH ERROR	7	T
DMA TST BAD DATA UNEXPECTED DMA INT UNEXPECTED IEEE-488 INT UNEXPECTED 60 Hz INT	8	E

The presence of a malfunction makes it necessary to turn off AC power at the system and remove the top cover from the 9520 chassis to examine the LEDs. Turn on AC power to repeat the self-test diagnostic and check the LEDs and the display. The LED associated with the failed task will be illuminated along with corresponding LEDs for any remaining tests that were interrupted. The cause of the malfunction will be indicated by the displayed error message and the missing characters in the C-O-M-P-L-E-T-E string.

At this point, rather than run system diagnostics, turn off AC power and check the printed circuit board connections to ensure the contacts are firmly seated in the back panel socket. Also, check the board interface harness and cable connectors to ensure that all connections are tight, as these connections can become loose if the equipment has been subjected to rough handling. Turn on AC power to repeat the diagnostic with board connections secured. Verify that the test is completed without malfunction, and that the message described in step 4 is displayed on the terminal screen to indicate that the system can be placed in operation as described in chapters 7 and 8.

If the malfunction is still present, it will be necessary to load the System Diagnostics diskette to isolate and correct the problem. Contact the Millennium System's Customer Service Representative via the telephone numbers listed in the preface.

Restarting System Operations

CAUTION: The RESET switch should not be activated during any system operation.

System operations can be restarted by pressing the RESET pushbutton switch located on the control panel. Operation of this switch terminates the current process operation and places the system in a reset state. This state causes the system to wait until the next command entry is issued from the terminal keyboard prior to beginning execution.

System Shutdown Operations

The system is shut down by first waiting for the current process operation to be completed so that the system is waiting for an input command from the keyboard. Next, open the disk access door (as described in Changing and Handling Diskettes) and remove the diskette. Operate the respective POWER switches on the development system control panel and at the display terminal to remove AC power from both units which completes the shutdown operation.

CHANGING AND HANDLING DISKETTES

Disk Drive Particulars

The two disk drives (figure 3-1) are mounted vertically on the front panel of the development system chassis. Drive A is located at the far left of the panel and Drive B is located near the center. Changing a diskette involves inserting and removing the diskette at a specified drive, and relocating a diskette from one drive to the other.

The latch bar on the drive access doors contain LED indicators which illuminate during system operations to show that data is being read from, or written to the diskette. These indicators must be turned off to show that read/write operations are completed before a diskete can be removed. Also, the system must be in the input mode (i.e. waiting for the next command entry from the keyboard before the diskette is removed.

```
**************************************************************
*                                                            *
*                        - CAUTION -                         *
*                                                            *
*  It is considered good practice to completely remove the   *
*  diskette from the drive, or to open the drive access      *
*  door (if the diskette is installed) prior to removing     *
*  and applying AC power to the system. This will avoid      *
*  possible disruption of recorded data which might occur    *
*  if the diskette is in contact with the drive spindle      *
*  during the power up/shutdown operation.                   *
*                                                            *
**************************************************************
```

Inserting Diskette in Drive

The diskette can be inserted in the disk drive with all power on and the drive spindle rotating. Use the following procedure (see figure 3-1).

1. Press the latch bar to open the access door at the selected drive. (The access door is spring-loaded to swing open when the latch bar is pressed.)

2. Position the diskette so that its Label Index Access Hole faces to the right (see figure 3-1) and the write protect tab is toward the top; then push the diskette into the opening as far as it will go.

3. Move the latch handle to the left to close the opening and lock the diskette on the drive spindle to complete the insertion procedure.

Removing Diskette from Drive

The diskette can be removed from the disk drive with all power on and the drive spindle rotating. Any read or write operation in process should be terminated before removing the diskette from the drive. Likewise, any data in memory that is to be saved should be written to the disk before it is removed. Use the following procedure (see figure 3-1):

1. Verify that all read/write operations have been completed at the selected drive (i.e. the LEDs on the latch handle should be turned off).

2. Press the latch bar to open the access door at the selected drive. (The access door is spring-loaded to swing open and eject the diskette from the spindle when the latch bar is pressed.)

3. Pull the diskette from the drive to complete the removal procedure.

Care of Diskette

The floppy diskette is a flexible disk enclosed in a permanent jacket. The interior of the jacket is lined with a wiping material which cleans the disk of foreign matter. the diskette should be stored in an envelope when it is removed from the drive.

Figure 3-1. Inserting Diskette in Drive

Handling and Storage: Special precautions should be used, as follows, to protect the diskette during handling and storing:

1. Return the diskette to its storage envelope whenever it is removed from the disk drive.

2. Keep diskette away from magnetic fields and from ferromagnetic materials which might become magnetized. Strong magnetic fields can distort recorded data on the diskette.

3. Replace storage envelopes when thy become worn, cracked or distorted. Envelopes are designed to protect the diskette.

4. Do not write on the plastic jacket with a lead pencil or ball-point pen. Use a felt tip pen.

5. Heat and contamination from a carelessly dropped tobacco ash can damage the diskette.

6. Do not expose diskette to heat or sunlight.

7. Do not touch or attempt to clean the diskette face. Abrasions may cause loss of stored data.

Write Protect Feature: The diskette has the capability of being write protected (i.e. writing to the diskette can be inhibited so that a read-only condition is present. This feature is particularly useful when a diskette contains master programs, text or other data intended for read-only purposes, and the need exits for preventing inadvertent writing to the disk that would destroy the data.

The write protect feature is enabled by the small hole located near the outside edge of the jacket (see figure 3-2). When the hole is open, writing is inhibited.

The capability for writing to the disk is enabled by covering the hole. The hole is covered by placing a paper tab over the front of the hole and folding the tab over the edge of the jacket to cover the rear of the hole. The tab can be removed from the hole at anytime to restore the write protect condition.

Figure 3-2. Diskette Write Protect Tab

THEORY OF OPERATION

GENERAL

The 9520 Microprocessor Software Development System is a software development system with 64K bytes of memory. This 64K bytes of memory is expandable to 112K bytes by the addition of an optional 48K-byte RAM board. The 9520 system also has two floppy disk drives that provide one megabyte of mass storage. Adding the optional memory converts the 9520 into a dual user system.

Operating the 9520 with the MP/M operating system allows the user to accomplish several software tasks simultaneously. For example, the user can assemble, edit, and spool to a printer, one or more text files.

The 9520 system provides a quick, easy human interface between the user and the computer. A full screen display furnishes access to large blocks of data at one time.

Interface between the 9520 Software Development System and peripherals is accomplished by three RS-232C ports, one high-speed serial RS-422 port and an eight-bit, parallel IEEE-488 port. These port configurations allow the 9520 to directly interface with a host computer and a hardware/software debug system providing a complete development system.

SYSTEM HARDWARE CONFIGURATION

The 9520 Software Development System hardware consists of five basic functional units (figure 4-1) that are located on a single, printed circuit board. The five units are as follows:

1. The central processing unit (CPU) is a Z80A microprocessor.

2. A 64K-byte dynamic RAM array, consisting of 32 2118 RAM chips. The RAM array is expandable to 112K-bytes with the 48K-byte option.

3. The direct memory access (DMA) and floppy disk controllers are Z80-DMA and WD1797 chips, respectively.

4. The two dual asynchronous receiver/transmitters (DARTs) are Z80-DART chips, that communicate with the four serial I/O ports.

5. The IEEE-488 port I/O controller is a 9914 GPIB device that provides 8-bit parallel communications with an 8-bit computer system.

The following paragraphs will highlight functions of the components comprising the specified basic units. The location for additional details pertinent to the individual units will be referenced in each discussion.

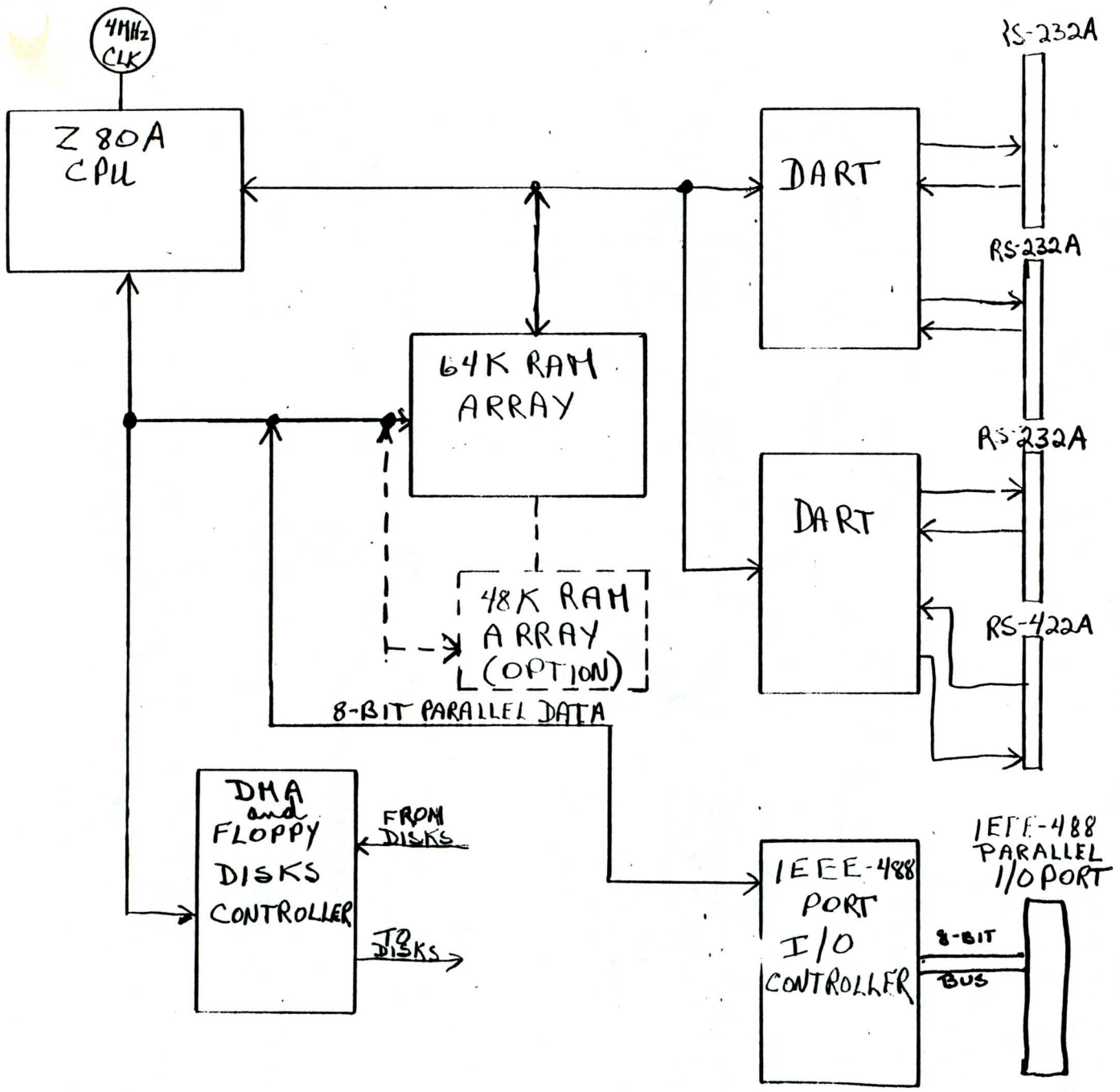


Figure 4.1. 4520 Printed Circuit Board Block Diagram

Z80A CPU

The Z80A microprocessor is the heart of the 9520 system. It functions to obtain instructions from memory and perform the requested operations. Memory is used to retain instructions and in most cases data that will be manipulated. For example, an instruction sequence may be read data from a peripheral device, store the data in a memory location, check parity, and write the data to another peripheral device. Thus, all necessary components are interfaced in a manner requiring minimal external logic. This CPU and its associated circuits enable the user to concentrate his efforts on software development. A brief description of the Z80A registers and their functions plus the instruction set by function are contained in this paragraph.

CPU Registers

Figure 4-2 illustrates how the registers are configured within the Z80A. These registers provide 208 bits of programmer accessible storage. There are two sets of 8-bit accumulators and flags, two sets of six 8-bit general purpose registers and six-special purpose registers, two that are 8-bit and four that are 16-bit.

Accumulator and Flag Registers

There are two independent 8-bit flag registers. The programmer selects the accumulator and flag pair with a single instruction. The accumulator holds the results of 8-bit arithmetic or logic operations and the flag register indicates specific conditions for 8- or 16-bit operations. Each of the two flag registers contains six bits of information that are set or reset by various CPU operations. Four of these bits - carry flag, zero flag, sign flag, and parity/overflow flag - are testable. The two non-testable bits, both used in BCD arithmetic, are half-carry and subtract flag. Two bits are unused.

General Purpose Registers

There are two matched sets of general purpose registers. Each set contains six 8-bit registers. These registers may be used individually as 8-bit registers or as 16-bit register pairs. One set of register pairs is called BC, DE and HL, the complementary set is called BC', DE' and HL'.

At any time, the programmer can select one or the other bank of general purpose registers. He can exchange one bank with the other bank by a single exchange command.

In systems where fast interrupt response is required, one set of general purpose registers, with an accumulator and flag register, may be reserved for fast routines. Only a simple exchange command needs to be executed to go between routines. This reduces interrupt service time by eliminating the requirement of saving and then retrieving register contents in the external stack during interrupt or subroutine processing.

Interrupt Vector (I)

The Z80 can be operated with an indirect call to any addressable memory location in response to an interrupt. The I register is used to store the high order 8-bits of the indirect address while the interrupting device provides the lower 8 bits of the address. This allows interrupt routines to be located throughout memory.

Memory Refresh (R)

This register allows the Z80 to automatically refresh dynamic memories. Seven bits of this memory refresh (R) register are incremented after each instruction fetch cycle. The eighth bit will remain as loaded with a LD R, A instruction. The data in the R register is sent out of the lower portion of the address bus (A0-A7) along with the $\overline{RFSH}$ (refresh) control signal while the Z80 is decoding and executing the instruction in the instruction register. The contents of the I (interrupt) register are placed on the upper 8 bits of the address bus (A8-A15).

The programmer can load the R register for testing purposes, but it is normally not used. The refresh is transparent to the user.

ACCUMULATOR A	FLAGS F	ACCUMULATOR A'	FLAGS F'	GENERAL PURPOSE REGISTERS
B	C	B'	C'	
D	E	D'	E'	
H	L	H'	L'	

Interrupt Vector I	Memory Refresh R	SPECIAL PURPOSE REGISTERS
Index Register 1X		
Index Register 1Y		
Stack Pointer SP		
Program Counter PC		

Figure A-2. Z80 Registers

IX AND IY (INDEX REGISTERS)

Each independent index register (IX and IY) holds a 16-bit base address used for indexed addressing. The index register is used as a base to point to a region in memory. An additional byte is included in all indexed instructions to specify a displacement from this base address. This displacement is specified as a two's complement signed integer.

STACK POINTER (SP)

The 16-bit address of the current top of a stack, located anywhere in the external RAM system memory, is held in the SP register. The external RAM system memory is organized as a last-in, first-out (LIFO) file. Data can be pushed onto the stack or popped off of the stack, using specific CPU registers, through the execution of PUSH or POP instructions. The data popped from the stack is always the last data pushed onto it.

PROGRAM COUNTER (PC)

The 16-bit address of the current instruction being fetched from memory is held in the PC. The PC is incremented after its contents have been transferred to the address lines. When a program jump occurs the new value is placed in the PC.

Z80A INSTRUCTION SUMMARY

The following pages contain a summary of the Z80A instruction set. The instructions are tabulated by function and are self-explanatory. Also included is a table showing the affect on flag information for the various Z80A instructions.

FLAGS

Table 4-1 shows how each flag is affected by Z80A instructions, tables 4-2 through 4-12 also provide flag information.

Each of the two Z80 flag registers have four testable bits, two non-testable bits and two unused bits. The four testable bits are:

1. Carry Flag (C) - This flag is the carry from the highest order bit of the accumulator. For example, the carry flag will be set during an add instruction where a carry from the highest bit of the accumulator is generated. This flag is also set if a borrow is generated during a subtraction instruction. Shift and rotate instructions also affect this flag.

2. Zero Flag (Z) - This flag is set if the result of the operation loaded a zero into the accumulator. This flag is also used with the block Search and block I/O instructions.

3. Sign Flag (S) - This flag stores the state of accumulator bit 7. This flag is intended to be used with signed numbers and it is set if the results of the operation was negative (bit 7 will be set to 1 with a negative number).

4. Parity/Overflow (P/V) - This is a dual purpose flag. It indicates the parity of the result in the accumulator when logical operations are performed (set to 1 = even parity). It indicates an overflow when a signed two's complement arithmetic operation exceeds the maximum possible (+127) or the minimum possible (-128).

The two non-testable bits in the flag register, both used for BCD arithmetic, are:

1. Half Carry (H) - This is the BCD carry or borrow result from the least significant four bits of operation. When using DAA (decimal adjust instruction) this flag is used to correct the result of a previous packed decimal add or subtract.

2. Subtract Flag (N) - Since the algorithm for correcting BCD operations is different for addition or subtraction, this flag is used to indicate what type of instruction was executed last so the DA operation will be correct for addition or subtraction.

Notice the relationship of the Z flag and the P/V flag with the block search, block transfer and block I/O instructions. This relationship is given in table 4-1 and is summarized below.

	Block Search	Block Transfer	Block I/O
Z=1	if A = (HL)	don't care	if B = 0
Z=0	if A = (HL)	don't care	if B = 0
P/V=1	if BC = 0	if BC = 0	don't care
P/V=0	if BC = 0	if BC = 0	don't care

When the block I/O transfer is complete, the Z flag will be set to one; when the block transfer is complete, the P/V flag is set to zero; when the block search is complete (fails to find a match) the P/V flag is set to zero.

The following pages contain tables 4-1 through 4-12, which provide a summary of the Z80A instruction set.

Table 4-1. Summary of Flag Operations

Instruction	C	Z	P/V	S	N	H	Comments
ADD A, s; ADC A, s	↕	↕	V	↕	0	↕	8-bit add or add with carry
SUB s; SBC A, s, CP s, NEG	↕	↕	V	↕	1	↕	8-bit subtract, subtract with carry, compare and negate accumulator
AND s	0	↕	P	↕	0	1	Logical operations
OR s, XOR s	0	↕	P	↕	0	0	And set's different flags
INC s	•	↕	V	↕	0	↕	8-bit increment
DEC m	•	↕	V	↕	1	↕	8-bit decrement
ADD DD, ss	↕	•	•	•	0	X	16-bit add
ADC HL, ss	↕	↕	V	↕	0	X	16-bit add with carry
SBC HL, ss	↕	↕	V	↕	1	X	16-bit subtract with carry
RLA; RLCA, RRA, RRCA	↕	•	•	•	0	0	Rotate accumulator
RL m;RLC m; RR m;RRC m SLA m; SRA m; SRL m	↕	↕	P	↕	0	0	Rotate and shift location m
RLD, RRD	•	↕	P	↕	0	0	Rotate digit left and right
DAA	↕	↕	P	↕	•	↕	Decimal adjust accumulator
CPL	•	•	•	•	1	1	Complement accumulator
SCF	1	•	•	•	0	0	Set carry
CCF	↕	•	•	•	0	X	Complement carry
IN r, (C)	•	↕	P	↕	0	0	Input register indirect
INI; IND; OUTI;OUTD	•	↕	X	X	1	X	Block input and output
INIR; INDR; OTIR; OTDR	•	1	X	X	1	X	Z = 0 if B ≠ 0 otherwise Z = 1
LDI, LDD	•	X	↕	X	0	0	Block transfer instructions
LDIR, LDDR	•	X	0	X	0	0	P/V = 1 if BC ≠ , otherwise P/V = 0
CPI, CPIR, CPD, CPDR	•	↕	↕	↕	1	X	Block search instructions Z = 1 if A = (HL), otherwise Z = 0 P/V = 1 if BC ≠ 0, otherwise P/V = 0
LD A, I; LD A, R	•	↕	IFF	↕	0	0	The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag
BIT b, s	•	↕	X	X	0	1	The state of bit b of location s is copied into the Z flag
NEG	↕	↕	V	↕	1	↕	Negate accumulator

The following notation is used in this table:

Symbol	Operation
C	Carry/link flag. C = 1 if the operation produced a carry from the MSB of the operand or result.
Z	Zero flag. Z = 1 if the result of the operation is zero.
S	Sign flag. S = 1 if the MSB of the result is one.
P/V	Parity or overflow flag. Parity (P) and overflow (V) share the same flag. Logical operations affect this flag with the parity of the result while arithmetic operations affect this flag with the overflow of this result. If P/V holds parity, P/V = 1 if the result of the operation is even, P/V = 0 if result is odd. If P/V holds overflow, P/V = 1 if the result of the operation produced an overflow.
H	Half-carry flag. H = 1 if the add or subtract operation produced a carry into or borrow from into bit 4 of the accumulator.
N	Add/Subtract flag. N = 1 if the previous operation was a subtract.
	H and N flags are used in conjunction with the decimal adjust instruction (DAA) to properly correct the result into packed BCD format following addition or subtraction using operands with packed BCD format.
↕	The flag is affected according to the result of the operation.
•	The flag is unchanged by the operation.
0	The flag is reset by the operation.
1	The flag is set by the operation.
X	The flag is a "don't care."
V	P/V flag affected according to the overflow result of the operation.
P	P/V flag affected according to the parity result of the operation.
r	Any one of the CPU registers A, B, C, D, E, H, L.
s	Any 8-bit location for all the addressing modes allowed for the particular instruction.
ss	Any 16-bit location for all the addressing modes allowed for that instruction.
ii	Any one of the two index registers IX or IY.
R	Refresh counter.
n	8-bit value in range <0, 255>
nn	16-bit value in range <0, 65535>
m	Any 8-bit location for all the addressing modes allowed for the particular instruction.

Table 4-2. 8-Bit Load Group

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments r, r'	Reg.
LD r, r'	r←r'	•	•	•	•	•	•	01 r r'	1	1	4		
LD r, n	r←n	•	•	•	•	•	•	00 r 110	2	2	7	000	B
								— n —				001	C
LD r, (HL)	r←(HL)	•	•	•	•	•	•	01 r 110	1	2	7	010	D
LD r, (IX+d)	r←(IX+d)	•	•	•	•	•	•	11 011 101	3	5	19	011	E
								01 r 110				100	H
								— d —				101	L
LD r, (IY+d)	r←(IY+d)	•	•	•	•	•	•	11 111 101	3	5	19	111	A
								01 r 110					
								— d —					
LD (HL), r	(HL)←r	•	•	•	•	•	•	01 110 r	1	2	7		
LD (IX+d), r	(IX+d)←r	•	•	•	•	•	•	11 011 101	3	5	19		
								01 110 r					
								— d —					
LD (IY+d), r	(IY+d)←r	•	•	•	•	•	•	11 111 101	3	5	19		
								01 110 r					
								— d —					
LD (HL), n	(HL)←n	•	•	•	•	•	•	00 110 110	2	3	10		
								— n —					
LD (IX+d), n	(IX+d)←n	•	•	•	•	•	•	11 011 101	4	5	19		
								00 110 110					
								— d —					
								— n —					
LD (IY+d), n	(IY+d)←n	•	•	•	•	•	•	11 111 101	4	5	19		
								00 110 110					
								— d —					
								— n —					
LD A, (BC)	A←(BC)	•	•	•	•	•	•	00 001 010	1	2	7		
LD A, (DE)	A←(DE)	•	•	•	•	•	•	00 011 010	1	2	7		
LD A, (nn)	A←(nn)	•	•	•	•	•	•	00 111 010	3	4	13		
								— n —					
								— n —					
LD (BC), A	(BC)←A	•	•	•	•	•	•	00 000 010	1	2	7		
LD (DE), A	(DE)←A	•	•	•	•	•	•	00 010 010	1	2	7		
LD (nn), A	(nn)←A	•	•	•	•	•	•	00 110 010	3	4	13		
								— n —					
								— n —					
LD A, I	A←I	•	↕	IFF	↕	0	0	11 101 101	2	2	9		
								01 010 111					
LD A, R	A←R	•	↕	IFF	↕	0	0	11 101 101	2	2	9		
								01 011 111					
LD I, A	I←A	•	•	•	•	•	•	11 101 101	2	2	9		
								01 000 111					
LD R, A	R←A	•	•	•	•	•	•	11 101 101	2	2	9		
								01 001 111					

Notes: r,r' means any of the registers A, B, C, D, E, H, L
IFF the content of the interrupt enable flip-flop (IFF) is copied into the P/V flag

Flag Notation: • = flag not affected. 0 = flag reset. 1 = flag set. X = flag is unknown.
↕ = flag is affected according to the result of the operation.

Table 4-3. 16-Bit Load Group

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
LD dd, nn	dd ← nn	•	•	•	•	•	•	00 dd0 001 – n – – n –	3	3	10	dd Pair 00 BC 01 DE 10 HL 11 SP
LD IX, nn	IX ← nn	•	•	•	•	•	•	11 011 101 00 100 001 – n – – n –	4	4	14	
LD IY, nn	IY ← nn	•	•	•	•	•	•	11 111 101 00 100 001 – n – – n –	4	4	14	
LD HL, (nn)	H ← (nn + 1) L ← (nn)	•	•	•	•	•	•	00 101 010 – n – – n –	3	5	16	
LD dd, (nn)	dd_H ← (nn + 1) dd_L ← (nn)	•	•	•	•	•	•	11 101 101 01 dd1 011 – n – – n –	4	6	20	
LD IX, (nn)	IX_H ← (nn + 1) IX_L ← (nn)	•	•	•	•	•	•	11 011 101 00 101 010 – n – – n –	4	6	20	
LD IY, (nn)	IY_H ← (nn + 1) IY_L ← (nn)	•	•	•	•	•	•	11 111 101 00 101 010 – n – – n –	4	6	20	
LD (nn), HL	(nn + 1) ← H (nn) ← L	•	•	•	•	•	•	00 100 010 – n – – n –	3	5	16	
LD (nn), dd	(nn + 1) ← dd_H (nn) ← dd_L	•	•	•	•	•	•	11 101 101 01 dd0 011 – n – – n –	4	6	20	
LD (nn), IX	(nn + 1) ← IX_H (nn) ← IX_L	•	•	•	•	•	•	11 011 101 00 100 010 – n – – n –	4	6	20	
LD (nn), IY	(nn + 1) ← IY_H (nn) ← IY_L	•	•	•	•	•	•	11 111 101 00 100 010 – n – – n –	4	6	20	
LD SP, HL	SP ← HL	•	•	•	•	•	•	11 111 001	1	1	6	
LD SP, IX	SP ← IX	•	•	•	•	•	•	11 011 101 11 111 001	2	2	10	
LD SP, IY	SP ← IY	•	•	•	•	•	•	11 111 101 11 111 001	2	2	10	
PUSH qq	(SP – 2) ← qq_L (SP – 1) ← qq_H	•	•	•	•	•	•	11 qq0 101	1	3	11	qq Pair 00 BC 01 DE 10 HL 11 AF
PUSH IX	(SP – 2) ← IX_L (SP – 1) ← IX_H	•	•	•	•	•	•	11 011 101 11 100 101	2	4	15	
PUSH IY	(SP – 2) ← IY_L (SP – 1) ← IY_H	•	•	•	•	•	•	11 111 101 11 100 101	2	4	15	
POP qq	qq_H ← (SP + 1) qq_L ← (SP)	•	•	•	•	•	•	11 qq0 001	1	3	10	
POP IX	IX_H ← (SP + 1) IX_L ← (SP)	•	•	•	•	•	•	11 011 101 11 100 001	2	4	14	
POP IY	IY_H ← (SP + 1) IY_L ← (SP)	•	•	•	•	•	•	11 111 101 11 100 001	2	4	14	

Notes: dd is any of the register pairs BC, DE, HL, SP
qq is any of the register pairs AF, BC, DE, HL
$(PAIR)_H$, $(PAIR)_L$ refer to high order and low order eight bits of the register pair respectively
E.g. BC_L = C, AF_H = A

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown
↕ = flag is affected according to the result of the operation

Table 4-4. Exchange Group and Block Transfer and Search Group

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
EX DE, HL	DE–HL	•	•	•	•	•	•	11 101 011	1	1	4	
EX AF, AF'	AF–AF'	•	•	•	•	•	•	00 001 000	1	1	4	
EXX	$\begin{pmatrix}BC\\DE\\HL\end{pmatrix}-\begin{pmatrix}BC'\\DE'\\HL'\end{pmatrix}$	•	•	•	•	•	•	11 011 001	1	1	4	Register bank and auxiliary register bank exchange
EX (SP), HL	$H-(SP+1)$ $L-(SP)$	•	•	•	•	•	•	11 100 011	1	5	19	
EX (SP), IX	$IX_H-(SP+1)$ $IX_L-(SP)$	•	•	•	•	•	•	11 011 101 11 100 011	2	6	23	
EX (SP), IY	$IY_H-(SP+1)$ $IY_L-(SP)$	•	•	•	•	•	•	11 111 101 11 100 011	2	6	23	
LDI	(DE)–(HL) DE–DE + 1 HL–HL + 1 BC–BC – 1	•	•	①	I	•	0	0	11 101 101 10 100 000	2	4	16	Load (HL) into (DE), increment the pointers and decrement the byte counter (BC)
LDIR	(DE)–(HL) DE–DE + 1 HL–HL + 1 BC–BC – 1 Repeat until BC = 0	•	•	0	•	0	0	11 101 101 10 110 000	2 2	5 4	21 16	If BC = 0 If BC ≠ 0
LDD	(DE)–(HL) DE–DE – 1 HL–HL – 1 BC–BC – 1	•	•	①	I	•	0	0	11 101 101 10 101 000	2	4	16	
LDDR	(DE)–(HL) DE–DE – 1 HL–HL – 1 BC–BC – 1 Repeat until BC = 0	•	•	0	•	0	0	11 101 101 10 111 000	2 2	5 4	21 16	If BC = 0 If BC ≠ 0
CPI	A – (HL) HL–HL + 1 BC–BC – 1	•	② I	① I	I	1	I	11 101 101 10 100 001	2	4	16	
CPIR	A – (HL) HL–HL + 1 BC–BC – 1 Repeat until A (HL) or BC = 0	•	② I	① I	I	1	I	11 101 101 10 110 001	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)
CPD	A – (HL) HL–HL – 1 BC–BC – 1	•	② I	① I	I	1	I	11 101 101 10 101 001	2	4	16	
CPDR	A – (HL) HL–HL – 1 BC–BC – 1 Repeat until a = (HL) or BC = 0	•	② I	① I	I	1	I	11 101 101 10 111 001	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)

Notes: ① P/V flag is 0 if the result of BC – 1 = 0, otherwise P/V = 1
② Z flag is 1 of A = (HL), otherwise Z = 0

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
I = flag is affected according to the result of the operation.

Table 4-5. 8-Bit Arithmetic and Logical Group

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
ADD A, r	A ← A + r	↕	↕	V	↕	0	↕	10 [000] r	1	1	4	r Reg.
ADD A, n	A ← A + n	↕	↕	V	↕	0	↕	11 [000] 110 — n —	2	2	7	000 B 001 C
ADD A,(HL)	A ← A + (HL)	↕	↕	V	↕	0	↕	10 [000] 110	1	2	7	010 D
ADD A,(IX + d)	A ← A + (IX + d)	↕	↕	V	↕	0	↕	11 011 101 10 [000] 110 — d —	3	5	19	011 E 100 H 101 L
ADD A,(IY + d)	A ← A + (IY + d)	↕	↕	V	↕	0	↕	11 111 101 10 [000] 110 — d —	3	5	19	111 A
ADC A, s	A ← A + s + CY	↕	↕	V	↕	0	↕	[001]				s is any of r, n, (HL), (IX + d), (IY + d) as shown for ADD instruction. The indicated bits replace the 000 in the ADD set above.
SUB s	A ← A − s	↕	↕	V	↕	1	↕	[010]				
SBC A, s	A ← A − s − CY	↕	↕	V	↕	1	↕	[011]				
AND s	A ← A ∧ s	0	↕	P	↕	0	1	[100]				
OR s	A ← A ∨ s	0	↕	P	↕	0	0	[110]				
XOR s	A ← A ⊕ s	0	↕	P	↕	0	0	[101]				
CO s	A − s	↕	↕	V	↕	1	↕	[111]				
INC r	r ← r + 1	•	↕	V	↕	0	↕	00 r [100]	1	1	4	
INC (HL)	(HL) ← (HL) + 1	•	↕	V	↕	0	↕	00 110 [100]	1	3	11	
INC (IX + d)	(IX + d) ← (IX + d) + 1	•	↕	V	↕	0	↕	11 011 101 00 110 [100] — d —	3	6	23	
INC (IY + d)	(IY + d) ← (IY + d) + 1	•	↕	V	↕	0	↕	11 111 101 00 110 [100] — d —	3	6	23	
DEC m	M ← m − 1	•	↕	V	↕	1	↕	[101]				m is any of r, (HL), (IX + d), (IY + d) as shown for INC. Same format and states as INC. Replaces 100 with 101 m OP code.

Notes: The V symbol in the P/V flag column indicates that the P/V flag contains the overflow of the result of the operation. Similarly the P symbol indicates parity. V = 1 means overflow, V = 0 means not overflow. P = 1 means parity of the result is even, P = 0 means parity of the result is odd.

Flag Notation: • = flag not affected. 0 = flag reset. 1 = flag set. X = flag is unknown.
↕ = flag is affected according to the result of the operation.

Table 4-6. General Purpose Arithmetic and CPU Control Groups

Mnemonic	Symbolic Operation	Flags						Op-Code	No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	P/V	S	N	H	76 543 210				
DAA	Converts acc. content into packed BCD following add or subtract with packed BCD operands	↕	↕	P	↕	•	↕	00 100 111	1	1	4	Decimal adjust accumulator
CPL	$A - \overline{A}$	•	•	•	•	1	1	00 101 111	1	1	4	Complement accumulator (one's complement)
NEG	A – 0 – A	↕	↕	V	↕	1	↕	11 101 101 01 000 100	2	2	8	Negate acc. (two's complement)
CCF	$CY - \overline{CY}$	↕	•	•	•	0	X	00 111 111	1	1	4	Complement carry flag
SCF	CY – 1	1	•	•	•	0	0	00 110 111	1	1	4	Set carry flag
NOP	No operation	•	•	•	•	•	•	00 000 000	1	1	4	
HALT	CPU halted	•	•	•	•	•	•	01 110 110	1	1	4	
DI	IFF – 0	•	•	•	•	•	•	11 110 011	1	1	4	
EI	IFF – 1	•	•	•	•	•	•	11 111 011	1	1	4	
IM 0	Set interrupt mode 0	•	•	•	•	•	•	11 101 101 01 000 110	2	2	8	
IM 1	Set interrupt mode 1	•	•	•	•	•	•	11 101 101 01 010 110	2	2	8	
IM 2	Set interrupt mode 2	•	•	•	•	•	•	11 101 101 01 011 110	2	2	8	

Notes: IFF indicates the interrupt enable flip-flop
CY indicates the carry flip-flop.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
↕ = flag is affected according to the result of the operation.

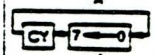
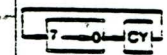
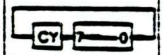
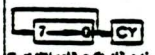
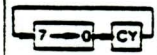
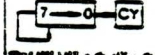
Table 4-7. 16-Bit Arithmetic Group

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
ADD HL, ss	HL ← HL + ss	:	•	•	•	0	X	00 ss1 001	1	3	11	ss Reg. 00 BC
ADC HL, ss	HL ← HL + ss + CY	:	:	V	:	0	X	11 101 101 01 ss1 010	2	4	15	01 DE 10 HL
SBC HL, ss	HL ← HL − ss − CY	:	:	V	:	1	X	11 101 101 01 ss0 010	2	4	15	11 SP
ADD IX, pp	IX ← IX + pp	:	•	•	•	0	X	11 011 101 00 pp1 001	2	4	15	pp Reg. 00 BC 01 DE 10 IX 11 SP
ADD IY, rr	IY ← IY + rr	:	•	•	•	0	X	11 111 101 00 rr1 001	2	4	15	rr Reg. 00 BC 01 DE 10 IY 11 SP
INC ss	ss ← ss + 1	•	•	•	•	•	•	00 ss0 011	1	1	6	
INC IX	IX ← IX + 1	•	•	•	•	•	•	11 011 101 00 100 011	2	2	10	
INC IY	IY ← IY + 1	•	•	•	•	•	•	11 111 101 00 100 011	2	2	10	
DEC ss	ss ← ss − 1	•	•	•	•	•	•	00 ss1 011	1	1	6	
DEC IX	IX ← IX − 1	•	•	•	•	•	•	11 011 101 00 101 011	2	2	10	
DEC IY	IY ← IY − 1	•	•	•	•	•	•	11 111 101 00 101 011	2	2	10	

Notes: ss is any of the register pairs BC, DE, HL, SP
pp is any of the register pairs BC, DE, IX, SP
rr is any of the register pairs BC, DE, IY, SP.

Flag Notation: • = flag not affected. 0 = flag reset. 1 = flag set. X = flag is unknown.
: = flag is affected according to the result of the operation.

Table 4-8. Rotate and Shift Group

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
RLCA		1	•	•	•	0	0	00 000 111	1	1	4	Rotate left circular accumulator
RLA		↕	•	•	•	0	0	00 010 111	1	1	4	Rotate left accumulator
RRCA		↕	•	•	•	0	0	00 001 111	1	1	4	Rotate right circular accumulator
RRA		↕	•	•	•	0	0	00 011 111	1	1	4	Rotate right accumulator
RLC r		↕	↕	P	↕	0	0	11 001 011 00 [000] -r	2	2	8	Rotate left circular register r
RLC (HL)		↕	↕	P	↕	0	0	11 001 011 00 [000] 110	2	4	15	r Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
RLC (IX + d)		↕	↕	P	↕	0	0	11 011 101 11 001 011 – d – 00 [000] 110	4	6	23	
RLC (IY + d)		↕	↕	P	↕	0	0	11 111 101 11 001 011 – d – 00 [000] 110	4	6	23	
RL m		↕	↕	P	↕	0	0	[010]				Instruction format and states are shown for RLC. m. To form new OP-code replace [000] of RLC. m with shown code
RRC m		↕	↕	P	↕	0	0	[001]				
RR m		↕	↕	P	↕	0	0	[011]				
SLA m		↕	↕	P	↕	0	0	[100]				
SRA m		↕	↕	P	↕	0	0	[101]				
SRL m		↕	↕	P	↕	0	0	[111]				
RLD		•	↕	P	↕	0	0	11 101 101 01 101 111	2	5	18	Rotate digit left and right between the accumulator and location (HL). The content of the upper half of the accumulator is unaffected
RRD		•	↕	P	↕	0	0	11 101 101 01 100 111	2	5	18	

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
↕ = flag is affected according to the result of the operation.

Table 4-9. Bit Set, Reset and Test Group

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
BIT b, r	$Z - \overline{r_b}$	•	↕	X	X	0	1	11 001 011 01 b r	2	2	8	r: 000 B, 001 C, 010 D, 011 E, 100 H, 101 L, 111 A
BIT b, (HL)	$Z - \overline{(HL)_b}$	•	↕	X	X	0	1	11 001 011 01 b 110	2	3	12	b: 000 Bit 0, 001 Bit 1, 010 Bit 2, 011 Bit 3, 100 Bit 4, 101 Bit 5, 110 Bit 6, 111 Bit 7
BIT b, (IX + d)	$Z - \overline{(IX + d)_b}$	•	↕	X	X	0	1	11 011 101 11 001 011 – d – 01 b 110	4	5	20	
BIT b, (IY + d)	$Z - \overline{(IY + d)_b}$	•	↕	X	X	0	1	11 111 101 11 001 011 – d – 01 b 110	4	5	20	
SET b, r	$r_b - 1$	•	•	•	•	•	•	11 001 011 [11] b r	2	2	8	
SET b, (HL)	$(HL)_b - 1$	•	•	•	•	•	•	11 001 011 [11] b 110	2	4	15	
SET b, (IX + d)	$(IX + d)_b - 1$	•	•	•	•	•	•	11 011 101 11 001 011 – d – [11] b 110	4	6	23	
SET b, (IY + d)	$(IY + d)_b - 1$	•	•	•	•	•	•	11 111 101 11 001 011 – d – [11] b 110	4	6	23	
RES b, m	$s_b - 0$ m = r, (HL), (IX + d), (IY + d)							[10]				To form new OP-code replace [11] of SET b, m with [10]. Flags and time states for SET instruction

Notes: The notation s_b indicates bit b (0 to 7) or location s.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown
↕ = flag is affected according to the result of the operation

Table 4-10. Jump Group

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
JP nn	PC – nn	•	•	•	•	•	•	11 000 011 – n – – n –	3	3	10	
JP cc, nn	If condition cc is true PC – nn, otherwise continue	•	•	•	•	•	•	11 cc 010 – n – – n –	3	3	10	
JR e	PC – PC + e	•	•	•	•	•	•	00 011 000 – e–2 –	2	3	12	
JR C, e	If C = 0, continue	•	•	•	•	•	•	00 111 000 – e–2 –	2	2	7	If condition not met
	If C = 1, PC – PC + e								2	3	12	If condition is met
JR NC e,	If C = 1, continue	•	•	•	•	•	•	00 110 000 – e–2 –	2	2	7	If condition not met
	If C = 0, PC – PC + e								2	3	12	If condition is met
JR Z, e	If Z = 0 continue	•	•	•	•	•	•	00 101 000 – e–2 –	2	2	7	If condition not met
	If Z = 1, PC – PC + e								2	3	12	If condition is met
JR NZ, e	If Z = 1, continue	•	•	•	•	•	•	00 100 000 – e–2 –	2	2	7	If condition not met
	If Z = 0, PC – PC + e								2	3	12	If condition met
JP (HL)	PC – HL	•	•	•	•	•	•	11 101 001	1	1	4	
JP (IX)	PC – IX	•	•	•	•	•	•	11 011 101 11 101 001	2	2	8	
JP (IY)	PC – IY	•	•	•	•	•	•	11 111 101 11 101 001	2	2	8	
DJNZ, e	B – B – 1 If B = 0, continue	•	•	•	•	•	•	00 010 000 – e–2 –	2	2	8	If B = 0
	If B = 0, PC – PC + e								2	3	13	If B = 0

cc	Condition	
000	NZ	non zero
001	Z	zero
010	NC	non carry
011	C	carry
100	PO	parity odd
101	PE	parity even
110	P	sign positive
111	M	sign negative

Notes: e represents the extension in the relative addressing mode.
e is signed two's complement number in the range < – 126, 129 >
e–2 is the op-code provides an effective address of pc + e as PC is incremented by 2 prior to the additin of e.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
↕ = flag is affected according to the result of the operation.

Table 4-11. Call and Return Group

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
CALL nn	$(SP-1)-PC_H$ $(SP-2)-PC_L$ $PC-nn$	•	•	•	•	•	•	11 001 101 – n – – n –	3	5	17	
CALL cc, nn	If condition cc is false continue, otherwise same as CALL nn	•	•	•	•	•	•	11 cc 100 – n –	3	3	10	If cc is false
								– n –	3	5	17	If cc is true
RET	$PC_L-(SP)$ $PD_H-(SP+1)$	•	•	•	•	•	•	11 001 001	1	3	10	
RET cc	If condition cc is false continue, otherwise same as RET	•	•	•	•	•	•	11 cc 000	1	1	5	cc is false
									1	3	11	if cc is true
RETI	Return from interrupt	•	•	•	•	•	•	11 101 101 01 001 101	2	4	14	
RETN	Return from non maskable interrupt	•	•	•	•	•	•	11 101 101 01 000 101	2	4	14	
RST p	$(SP-1)-PC_H$ $(SP-2)-PC_L$ PC_H-0 PC_L-P	•	•	•	•	•	•	11 t 111	1	3	11	

cc	Condition	
000	NZ	non zero
001	Z	zero
010	NC	non carry
011	C	carry
100	PO	parity odd
101	PE	parity even
110	P	sign positive
111	M	sign negative

t	P
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
: = flag is affected according to the result of the operation.

Table 4-12. Input and Output Group

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
IN A, (n)	A←(n)	•	•	•	•	•	•	11 011 011 ← n →	2	3	11	n to A_0 – A_7 Acc to A_8 – A_{15}
IN r, (C)	r←(C) if r = 110 only the flags will be affected	•	↕	P	↕	0	↕	11 101 101 01 r 000	2	3	12	C to A_0 – A_7 B to A_8 – A_{15}
INI	(HL)←(C) B←B – 1 HL←HL + 1	•	↕ ①	X	X	1	X	11 101 101 10 100 010	2	4	16	C to A_0 – A_7 B to A_8 – A_{15}
INIR	(HL)←(C) B←B – 1 HL←HL + 1 Repeat until B = 0	•	1 ①	X	X	1	X	11 101 101 10 110 010	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to A_0 – A_7 B to A_8 – A_{15}
IND	(HL)←(C) B←B – 1 HL←HL – 1	•	↕ ①	X	X	1	X	11 101 101 10 101 010	2	4	16	C to A_0 – A_7 B to A_8 – A_{15}
INDR	HL←(C) B←B – 1 HL←HL – 1 Repeat until B = 0	•	1 ①	X	X	1	X	11 101 101 10 111 010	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to A_0 – A_7 B to A_8 – A_{15}
OUT (n), A	(n)←A	•	•	•	•	•	•	11 010 011 ← n →	2	3	11	n to A_0 – A_7 Acc to A_8 – A_{15}
OUT (C), r	(C)←r	•	•	•	•	•	•	11 101 101 01 r 001	2	3	12	C to A_0 – A_7 B to A_8 – A_{15}
OUTI	(C)←(HL) B←B – 1 HL←HL + 1	•	↕ ①	X	X	1	X	11 101 101 10 100 011	2	4	16	C to A_0 – A_7 B to A_8 – A_{15}
OTIR	(C)←(HL) B←B – 1 HL←HL + 1 Repeat until B = 0	•	1 ①	X	X	1	X	11 101 101 10 110 011	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to A_0 – A_7 B to A_8 – A_{15}
OUTD	(C)←(HL) B←B – 1 HL←HL – 1	•	↕ ①	X	X	1	X	11 101 101 10 101 011	2	4	16	C to A_0 – A_7 B to A_8 – A_{15}
OTDR	(C)←(HL) B←B – 1 HL←HL – 1 Repeat until B = 0	•	1 ①	X	X	1	X	11 101 101 10 111 011	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to A_0 – A_7 B to A_8 – A_{15}

Notes: ① If the result of B – 1 is zero the Z flag is set, otherwise it is reset.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
↕ = flag is affected according to the result of the operation.

64K-BYTE DYNAMIC RAM MEMORY ARRAY

The 64K-byte dynamic RAM consists of 36 16K x 1 bit, 2118 RAM chips. This RAM array is configured in 4 rows and eight data columns, plus an additional four (1 row) 2118 RAM chips that provide an odd parity check for the memory array. Memory access is provided by an internal controller that selects the row and column requestd by the CPU. The 9520 common operating system memory is located at addresses C000H through FFFFH (16K bytes) (see figure 4-3). User memory is located at addresses 0000H through BFFFH (48K bytes). The optional 48K bytes of additional memory are located in a second memory bank also addressed at locations 0000H through BFFFH. Bank selection is determined by the bank select I/O port address F8H. The bank select I/O port bit configuration is: B0 = CPU bank select, B4 = DMA bank select. Bits 1, 2, 3, 5, 6, and 7 are not used and are reserved for future use.

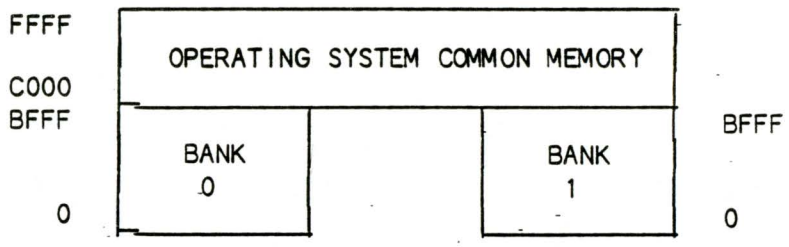


Figure 4-3. Operating and Bank Memories

FLOPPY DISK CONTROLLER AND DMA CONTROLLER

Floppy Disk Controller

The floppy disk controller is a Western Digital 1797 device that contains all the circuits necessary to provide the interface between the microprocessor and the floppy diskettes. The 1797 device performs parallel-to-serial and serial-to-parallel conversions, CRC generation and checking, the stepper motor interface and write procompensation, as well as recognizing interrupts and DMA requests. Associated circuits provide a VCO data separator. The controller is capable of reading and writing both single-density FM and double-density MFM encoded diskettes. The floppy disk controller has seven registers, which are used for the operation of the device, as shown in table 4-13. Tables 4-14 through 4-20 provide status register and command register summaries. For additional data refer to Western Digital Corporation Data Sheet.

Table 4-13. Floppy Disk Controller Registers

REGISTER HEX ADDRESS	DATA DIRECTION	REGISTER NAME
28	INPUT	STATUS
29	INPUT/OUTPUT	TRACK
2A	INPUT/OUTPUT	SECTOR
2B	INPUT/OUTPUT	DATA
28	OUTPUT	COMMAND

Table 4-14. Status Register Summary

BIT	ALL TYPE I COMMANDS	READ ADDRESS	READ SECTOR	READ TRACK	WRITE SECTOR	WRITE TRACK
S7	NOT READY	NOT READY	NOT READY	NOT READY	NOT READY	NOT READY
S6	WRITE PROTECT	0	0	0	WRITE PROTECT	WRITE PROTECT
S5	HEAD LOADED	0	RECORD TYPE	0	WRITE FAULT	WRITE FAULT
S4	SEEK ERROR	RNF	RNF	0	RNF	0
S3	CRC ERROR	CRC ERROR	CRC ERROR	0	CRC ERROR	0
S2	TRACK 0	LOST DATA	LOST DATA	LOST DATA	LOST DATA	LOST DATA
S1	INDEX	DRQ	DRQ	DRQ	DRQ	DRQ
S0	BUSY	BUSY	BUSY	BUSY	BUSY	BUSY

Table 4-15. Status for Type I Commands

BIT NAME	MEANING
S7 NOT READY	This bit when set indicates the drive is not ready. When reset it indicates that the drive is ready. This bit is an inverted copy of the Ready input and logically 'ored' with MR.
S6 PROTECTED	When set, indicates Write Protect is activated. This bit is an inverted copy of $\overline{\text{WRPT}}$ input.
S5 HEAD LOADED	When set, indicates the head is loaded and engaged. This bit is a logical "and" of HLD and HLT signals.
S4 SEEK ERROR	When set, the desired track was not verified. This bit is reset to 0 when updated.
S3 CRC ERROR	CRC encountered in ID field.
S2 TRACK 00	When set, indicates Read/Write head is positioned to Track 0. This bit is an inverted copy of the $\overline{\text{TR00}}$ input.
S1 INDEX	When set, indicates index mark detected from drive. This bit is an inverted copy of the $\overline{\text{IP}}$ input.
S0 BUSY	When set command is in progress. When reset no command is in progress.

Table 4-16. Status for Type II and III Commands

BIT NAME	MEANING
S7 NOT READY	This bit when set indicates the drive is not ready. When reset, it indicates that the drive is ready. This bit is an inverted copy of the Ready input and 'ored' with MR. The Type II and III Commands will not execute unless the drive is ready.
S6 WRITE PROTECT	On Read Record: Not Used. On Read Track: Not Used. On any Write: It indicates a Write Protect. This bit is reset when updated.
S5 RECORT TYPE/ WRITE FAULT	On Read Record: It indicates the record-type code from data field address mark. 1 = Deleted Data Mark. 0 = Data Mark. On any Write: It indicates a Write Fault. This bit is reset when updated.
S4 RECORD NOT FOUND (RNF)	When set, it indicates that the desired track, sector, or side were not found. This bit is reset when updated.
S3 CRC ERROR	If S4 is set, an error is found in one or more ID fields; otherwise it indicates error in data field. This bit is reset when updated.
S2 LOST DATA	When set, it indicates the computer did not respond to DRQ in one byte time. This bit is reset to zero when updated.
S1 DATA REQUEST	This bit is a copy of the DRQ output. When set, it indicates the DR is full on a Read Operation or the DR is empty on a Write operation. This bit is reset to zero when updated.
S0 BUSY	When set, command is under execution. When reset, no command is under execution.

Table 4-17. Command Summary

		BITS							
TYPE	COMMAND	7	6	5	4	3	2	1	0
I	Restore	0	0	0	0	h	V	r_1	r_0
I	Seek	0	0	0	1	h	V	r_1	r_0
I	Step	0	0	1	u	h	V	r_1	r_0
I	Step In	0	1	0	u	h	V	r_1	r_0
I	Step Out	0	1	1	u	h	V	r_1	r_0
II	Read Sector	1	0	0	m	F_2	E	F_1	0
II	Write Sector	1	0	1	m	F_2	E	F_1	a_0
III	Read Address	1	1	0	0	0	E	0	0
III	Read Track	1	1	1	0	0	E	0	0
III	Write Track	1	1	1	1	0	E	0	0
IV	Force Interrupt	1	1	0	1	I_3	I_2	I_1	I_0

Table 4-18. Flag Summary

TYPE I COMMANDS

h = Head Load Flag (Bit 3)
- h = 1, Load head at beginning
- h = 0, Unload head at beginning

V = Verify flag (Bit 2)
- V = 1, Verify on destination track
- V = 0, No verify

r_1r_0 = Stepping motor rate (Bits 1-0)
Refer to Table 1 for rate summary

u = Update flag (Bit 4)
- u = 1, Update Track register
- u = 0, No update

Table 4-19. Flag Summary

TYPE II & III COMMANDS

m = Multiple Record flag (Bit 4)
- m = 0, Single Record
- m = 1, Multiple Records

a_0 = Data Address Mark (Bit 0)
- a_0 = 0, FB (Data Mark)
- a_0 = 1, F8 (Deleted Data Mark)

E = 15 ms Delay (2 MHz)
- E = 1, 15 ms delay
- E = 0, no 15 ms delay

(F_2) S = Side Select Flag (1791/3 only)
- S = 0, Compare for Side 0
- S = 1, Compare for Side 1.

(F_1) C = Side Compare Flag (1791/3 only)
- C = 0, disable side select compare
- C = 1, enable side select compare

(F_1) S = Side Select Flag (Bit 1, 1795/7 only)
- S = 0 Update SSO to 0
- S = 1 Update SSO to 1

(F_2) b = Sector Length Flag (Bit 3, 1975/7 only)

	Sector Length Field			
	00	01	10	11
b = 0	256	512	1024	128
b = 1	128	256	512	1024

Table 4-20. Flag Summary

TYPE IV COMMANDS
Ii = Interrupt Condition flags (Bits 3-0) I0 = 1, Not-Ready to Ready Transition I1 = 1, Ready to Not-Ready Transition I2 = 1, Index Pulse I3 = 1, Immediate Interrupt I_3-I_0 = 0, Terminate with no Interrupt

DMA Controller

The Z80-DMA circuit is a programmable, single channel device that provides all address, timing, and control signals to effect the transfer of blocks of data between two ports within the Z80-CPU based system. These ports may be system main memory or any system peripheral I/O device. The DMA can also search a block of data for a particular byte (bit maskable), with or without a simultaneous transfer. The address of the DMA controller port is 18H. For additional data on the DMA controller, refer to the Zilog Corporation or Mostek Corporation Data Books for the Z80-DMA device.

Z80-Dual Asynchronous Receiver/Transmitters (DARTS)

The two Z80-DART devices provide the serial I/O interface between the 9520 Software Development System, the three RS-232-C ports and the one RS-422 port. The DARTs are dual asynchronous format controllers capable of operation up to 19,200 baud for the RS-232 ports. The four ports connected to the DARTs support the primary and secondary handshake signals necessary to prevent data overrun. These DARTs operate in the Z80 CPU mode 2 interrupt structure and supply vectored interrupts for character transmission, reception and changes in status signals. Each DART has nine registers, which control the two ports. Higher registers are addressed by the lower 3 bits of register 0. Register name, function and address for each of the RS-232 port pairs are listed in table 4-21:

Table 4-21. RS-232 Port Addresses and Functions

REGISTER HEX ADDRESS	DIRECTION	REGISTER
02	I/O	REM(ote)I/O data
03	I/O	REM(ote)I/O read register 0
08	I/O	RDR/PCH DATA
09	I/O	RDR/PCH read register 0
0A	I/O	Console data
0B	I/O	Console read/write register 0

449

The high-speed RS-422 serial port is capable of speeds up to 750,000 baud. The port assignments are as follows:

HEX ADDRESS	DIRECTION	REGISTER NAME
00	I/O	RS-422 data
01	I/O	RS-422 read/write register 0

Additional data pertinent to the Z-80 DART devices is available in Zilog Corporation Z-80 SIO Technical Manual.

IEEE-488 PORT I/O CONTROLLER

The 9520 system provides a high-speed, general purpose interface bus (GPIB). This GPIB, 8-bit parallel interface enhances the system by permitting communication with any similarly configured system. The GPIB interface is interrupt driven, similarly to the serial I/O ports. The GPIB port is controlled by a Texas Instrument 9914 that handles the bus protocol for command and data transfer. Switch S6, on the system rear panel, is read at power-on or reinitialization and assigns the peripheral address for port communication. The 9914 device has the capability of being a talker, listener, or controller. The interface drivers between the controller and the physical GPIB bus are configured for tristate operation, permitting a higher data transfer speed. The 9914 controller has 14 registers that control the device, table 4-15 provides the register hex address, register name and direction.

Table 4-22. 9914 Registers

REGISTER HEX ADDRESS	DIRECTION	REGISTER NAME
30	I/O	Interrupt Status 0 / Mask 0
31	I/O	Interrupt Status 1 / Mask 1
32	I	Address Status
33	I/O	Bus Status/Auxilliary Command
34	I/O	Address Switch/Address Register
35	O	Serial Poll
36	I/O	Command Pass Thru/Parallel Poll
37	I/O	Data

Additional information relative to the IEEE-488 Port I/O Controller is available in the Texas Instruments, Inc. TMS9914 GPIB Adapter Data Manual.

DISK DRIVES

The 9520 Software Development System contains two eight-inch floppy disk drives installed within the cabinet. All circuits necessary to interface and interpret the disk drive inputs and outputs are on the CPU processor board.

SOFTWARE DESCRIPTION

GENERAL

Software for the 9520 Development System is based upon the MP/M Operating System. MP/M incorporates command formats, conventions, syntax and file structures that support the following types of programming capabilities:

- Text editor to prepare and manipulate the users source program files.
- A variety of assemblers that translate source program files into target object files for different types of microprocessors.
- Linker to link object programs together to produce the executable load module.
- Utility programs to download and upload executable load modules between the 9520 Development System and a remote debug/emulator station.
- Floppy disk utility to format, duplicate and copy information on diskettes.
- I/O Baud Rate utility to display and set baud rate values for the 9520 I/O ports.

The 9520 Software is contained on *three system diskettes:

(1) System Diskette, Part #XXXXXXXX

(2) *Language Translator Diskette, Part #XXXXXXXX

(3) System Diagnostic Diskette, Part #XXXXXXXX

*NOTE: Separate Language Translator diskettes are available for the various assemblers and cross assemblers that are used to support the preparation of object files for different types of microprocessors.

SOFTWARE ORGANIZATION

The 9520 Development System software is organized around the operating system and applications software as shown in figure 5-1. The operating system is described in the MP/M Users Manual. The applications software can be divided into two functional groups: (1) Host Support Functions, and (2) Cross Support Functions.

Software Support Functions

Host support functions contain all program files that are made available to the 9520. Cross support functions contain specific program files that can be accessed by a remote debug/emulator station to establish communications with, and issue commands to the host system for upload, download and conversion operations as shown in figure 5-2.

Both the host support and cross support software is further divided into memory and disk resident programs. This distinction (memory/disk) is merely to differentiate between programs that are permanently resident in the system memory from those that are accessed from the disk. Programs that reside in memory are classified as program relocatable modules (PRLs). The PRLs associated with the host support and cross support functions are shown in figure 5-2.

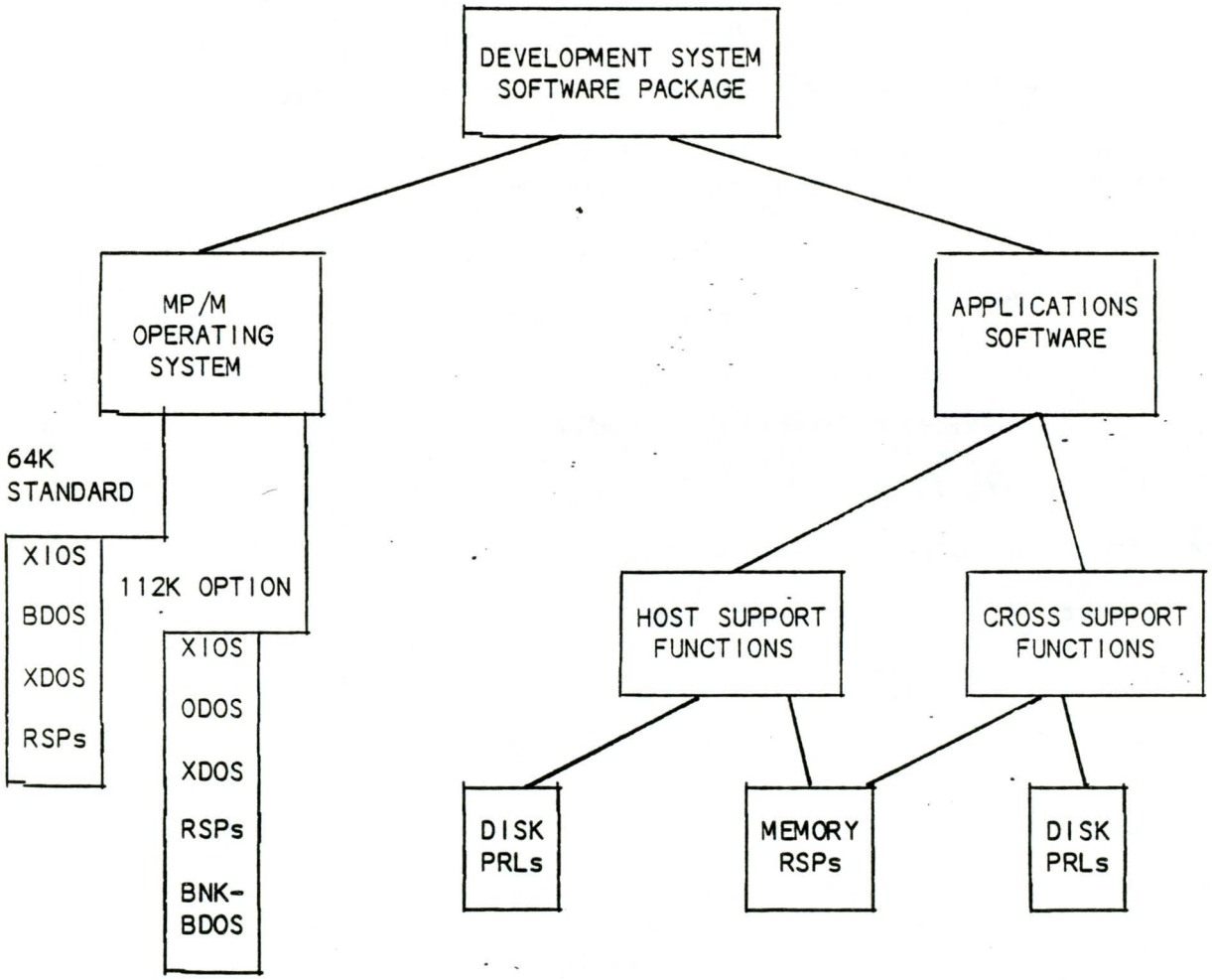


Figure 5-1. Development System Software Organization

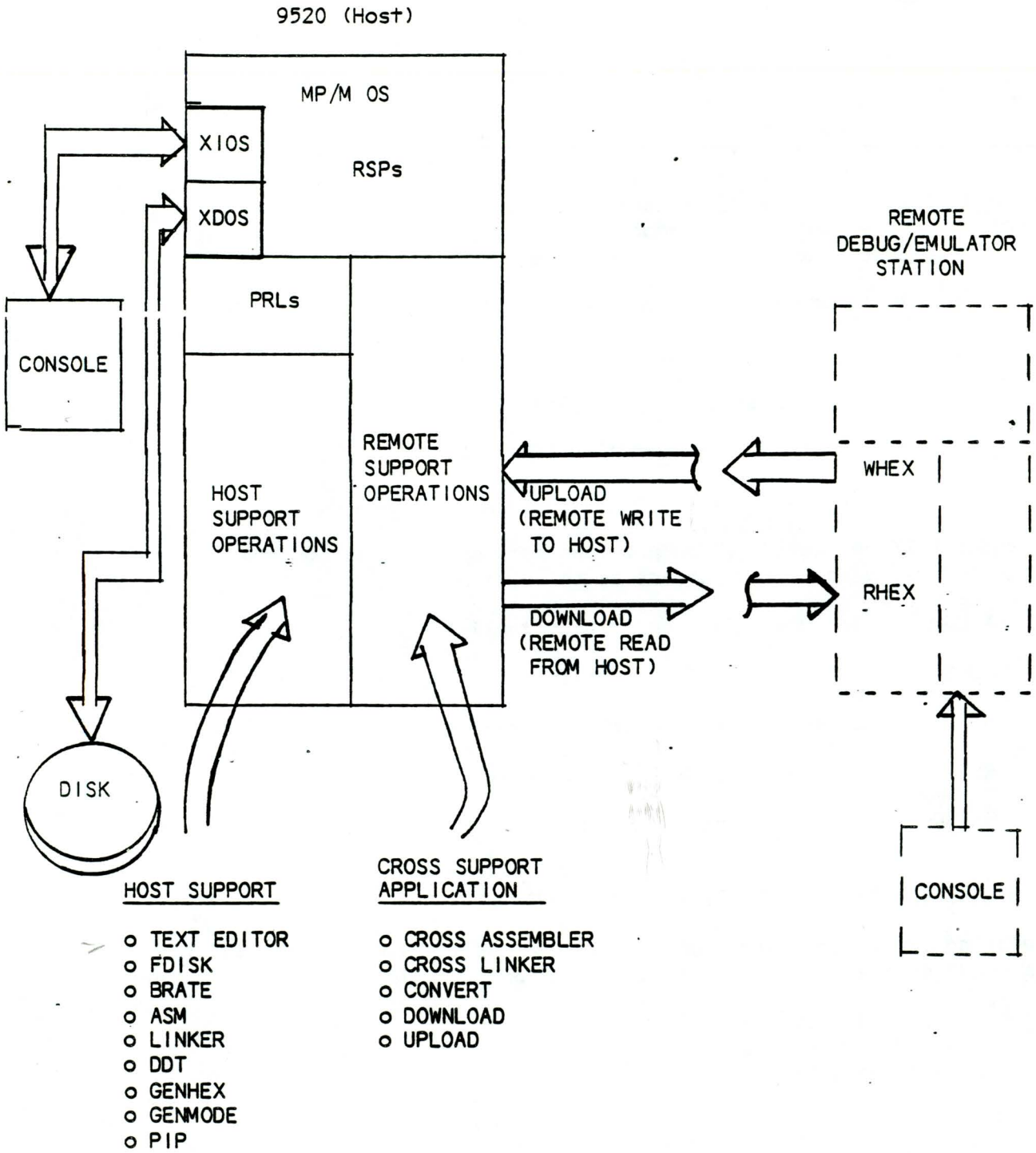


Figure 5-2. Cross Support Interface with Remote Station

Operating System Generation

The resident system processes are configured by the user when the system is initialized for system generation (GENSYS). The procedure for using the GENSYS program is described in chapter 7. The GENSYS parameters are loaded from the system diskette whenever the system is powered up or reset. These parameters define the software operating characteristics that differentiate one system from another. For example, which resident processes can be available, the amount of memory segment space provided, the number of consoles assigned, etc. The allocation of memory segments to accommodate the various operating system configurations as described in the MP/M Users Manual.

The standard, 64K memory system is configured with the following modules:

XIOS -- Basic and Extended I/O System File

BDOS -- Basic Disk Operating System File

XDOS -- Extended Disk Operating System File

RSPs -- Resident System Processes File

The expanded 112K memory operation provides more space to accomodate the extended processing modules which are as follows:

XIOS -- Basic and Extended I/O System File

ODOS -- BDOS with resident portion of BNKBDOS File

XDOS -- Extended Disk Operating System File

RSPs -- Resident System Processes File

BNKBDOS -- Bank Switched BDOS File

FUNCTIONAL DESCRIPTION OF PROCESSING FUNCTIONS

The operating system is based upon a real-time, multi-tasking nucleus. The nucleus provides the following software function controls:

- Process Dispatching
- Queue Management
- Flag Management
- Memory Management
- System Timing Functions

Process Dispatching Functions

Each process in the system has a process descriptor which contains all of the information needed by the system to define a process. This information is used during dispatching to save the state of the process that is currently being run, to determine which process is to be run, and then to restore that processing state.

The process descriptor contains all primitives for manipulation of system queues (Task Control Blocks) that are waiting for some event to occur. The Task Control Block (TCB) is actually an executing program. It is not the program itself that makes a task, but rather, its execution on the processor. The same program can be executed by any number of tasks, each with its own TCB. A task is therefore the state of executing a body of code.

Process dispatching is performed at each system call, at each interrupt and at each clock pulse of the system clock. Processes containing the same priority are scheduled for equal slices of CPU time.

The primary purpose of the process dispatcher is to assign the CPU to the next available task. When the task becomes available, it is given control of the CPU by the process dispatcher. The task maintains control of the CPU until it either gives up the control voluntarily, or requests an operating system service function that causes the task to wait for some event to occur. In either case, the tasks operating registers are saved in the TCB, the task is placed on an appropriate queue and control is returned to the process dispatcher to look for the next task top dispatch.

Queue Management Functions

Queues perform several critical functions in a real-time multi-tasking environment. The queues are used for controlling the order in communicating messages between processes, mutual exclusion of processes and to synchronize processes for the order, time and priority of servicing. The queue controls a first-in first-out list of messages, a list of processes awaiting messages, and the sequencing for sending each message.

Flag Management Functions

Flag Management is used to synchronize processes by signaling a specific event. The flags provide a logical interrupt system which is independent of the physical interrupt system. The flag interrupt therefore maps an arbitrary physical interrupt environment into a dynamic structure.

Memory Management Functions

Memory is managed by pre-defined segments. The MP/M System can accept up to eight memory segments of 48K. The extended 48K memory option of the 9520 Development System is supported by the banked-switched, memory allocation feature.

System Timing Functions

System Timing functions provide the time-of-day clock in addition to the capability to schedule programs that are loaded into memory from the disk to be executed. The timing functions also provide a capability to delay the execution of a process within a specified period of time. A special command can be issued at the console to read and set the date and time.

CONSOLE COMMANDS

This section describes the commonly used console commands that are issued by the operator to communicate with the software. The commands available at the console are established by resident system process (RSP) programs that are specified by the user during system generation (see chapter 7) and by program relocatable (PRL) modules that reside on the diskette.

The MP/M system does not use a system-defined, or built-in command structure that conatins a fixed number of commands. Each available command coincides with a specific RSP or PRL program that is configured by the user during the system generation process. These commands can be categorized into four groups as follows:

1. User Identification Commands
2. File Manipulation Commands
3. System Operation Commands
4. Program Operation Commands

Details of individual commands associated with each category are described in subsequent paragrapsh and the MP/M Users Guide Manual that is shipped with the system.

USER IDENTIFICATION COMMANDS

The user identification commands permit a user to perform the following functions:

- o Display current user ID code and/or set a user code value
- o Display user console ID number
- o Reset Disk Drives

Command Name: GET/SET USER CODE

Function: The GET/SET USER CODE Command is used to display the current user code as well as to set a user code value.

The command is invoked as follows:

To Display User Code: Entering the command, USER, followed by a <cr> will display the current user code. (Observe that the current user code is always displayed in the prompt.)

```
Entry:                  0A> USER cr
Syntax:                 0A> <USER> <cr>

System Response:        USER = 0
```

To Set the User Code to a Specified Number: Entering the command, USER, followed in sequence by a space, user code and then a <cr> will set the user code to the specified user code number. The example which follows assigns a user code value of 2

```
Entry:                  0A> USER 2 cr
Syntax:                 0A> <USER> <z> <cr>

System Response:        USER = 2

                        2A>
```

where: Legal user code numbers must be within the range of 0 - 15.

Command Name: CONSOLE

Function: The CONSOLE command is used to display the console ID number at the location where the command is being entered. The console number thus displayed, allows the user to examine system status to determine which processes are detached from consoles:

The command is invoked as follows:

```
Entry:                  0A CONSOLE cr
Syntax:                 0A> <CONSOLE> <cr>

System Response:        CONSOLE = 0
```

where: Console = 0 is displayed to indicate the assigned console ID number.

Command Name: DSKRESET

Function: The DSKRESET (Disk Reset) command is used to allow the operator to change disks. If there are open files on any of the drives to be reset, the disk reset is not allowed, and the cause of the rejection is displayed in an error message. Open files (i.e., in processof being written to the diskette) will lose their updated information if they are not closed prior to issuing the disk reset command.

The command is invoked as follows:

To reset all drives: Entering the DSKRESET command followed by a <cr> allows all drives to be reset:

Entry: 0A DSKRESET cr
Syntax: 0A> <DSKRESET> <cr>

Error Message: If a file is open on a drive, the following message is displayed:

Disk reset denied, Drive x: Console y Program z

where: x = Drive where program is being executed
y = Console where program is attached
z = Program Name that is still executing

To reset a specific drive: Entering the DSKRESET command, followed in sequence by a space, the drive descripter and a <cr> allows a specific drive to be reset. The following example describes how to reset Drive B:

Entry: 0A> DSKRESET B: cr
Syntax: 0A> <DSKRESET> <B:> <cr>

where: DSKRESET = Disk reset command
B = Drive descriptor with semicolon to denote I/O device

Error Message: Same as given for Reset all Drives.

FILE MANIPULATION COMMANDS

The File Manipulation Commands allow the user to perform the following operations on system files:

- Erase a File
- Display Contents of Source File
- List all Filenames
- Change Filenames
- Provide Status of File Storage
- Produce Hexadecimal Type Files from COM type files
- Produce COM Type Files from PRL Type files

The parameter that is entered in a command to reference the filename consists of two parts, primary name and secondary name (which is an extension of the primary name). The primary name is a 1-8 character, user-assigned name for the file. The secondary name is a 1-3 character extension name for the file type that may be assigned by the user or, under certain circumstances, a utility that is assigned by default.

A period (.) is used to separate the primary filename from the secondary extension, and is used in a command in the following forms to specify the type of processing search that will be performed on files:

where: Variations of the parameter entry are:

<*> <.*> = Process all system files (filename and extension parameters do not define a specific filename or file type).
<*> <.ASM> = Process only those files that are assigned with the ASM file type descriptor.
<W> <.ASM> = Process only the file that is referenced by the W filename and ASM file type descriptors.
<W> <.*> = Process only those files that are assigned the W filename descriptor.

Command Name: ERASE FILE

Function: Two commands, ERA (erase) and ERAQ (erase Query), are used to delete files. The ERA command is an immediate entry which allows the user to delete a specified file or all files. The ERAQ command is an interactive entry which initially displays a list of specified files in the system; the user is prompted by a query to respond with Y or N (for a yes or no confirmation) to delete each file contained in the list.

To delete all files: Entering the ERA command followed in sequence by two parameters (*.*) for filename and extension, and a <cr> will delete all files assigned for the user code.

Entry: 0A> ERA *.* cr
Syntax: 0A> <ERA> <*> <.*> cr

where: ERA = Erase Command

* = first parameter (required) which defines 8 characters or less for all primary filenames.
.* = second parameter (required) which defines 3 characters or less for all secondary filename extensions.
cr = carriage return (required)

System Response Message: Confirm delete all user files (Y/N)?

where: User must type a Y (for yes) or N (for no) in order to delete all files.

To display list of all files and selectively delete various filename(s) from list: Entering the ERAQ command followed by two parameters for filename and extension (*.*) causes the system to display the list for all fiulenames contained in the directory; the user is then prompted to respond with a Y (for yes) or N (for no) to delete or save each filename as it appears in the display.

Entry: 0A> ERAQ *.* cr
Syntax: 0A> <ERAQ> <*> <.*> <cr>

where: Definition of parameters is given in previous command.

To display specific files and delete from list: Entering the ERAQ command followed by the specified filename parameters allows the user to delete only those files that match the filename reference.

```
Entry:                  0A> ERAQ *.LST cr
Syntax:                 0A> <ERAQ> <*> <.LST> <cr>
```

where: ERAQ = command for selective deletion of file.
* = first parameter (required) which defines 8 characters or less for primary filenames.
.LST = second parameter (required) which defines 3 characters or less for the secondary filename (extension) of each file associated with the .LST parameter.
cr = carriage return.

```
System Response:        A:XIOS LST? Y
                        A:MYFILE LST? N
```

where: The selected files are displayed with a query requesting the user to respond with a Y (yes) or N (no) to delete or save the specified file.

Error Message: The message, NO FILE, is displayed if the filename cannot be found on the specified diskette.

Command Name: TYPE A FILE

Function: The TYPE command displays the contents of a specified ASCII source file on the screen. The user can specify the number of lines of data to be displayed on a page. The TYPE command automatically expands tabs at every eighth-column location.

The filename must be specified with two parameters following the TYPE command. The TYPE command has an optional pause mode to halt the display after the specified number of lines appear on the screen. The pause mode is set by entering a P, followed by two decimal digits (to indicate line count) after the filename extension entry. The pause mode will cause the display to halt until the <cr> key is pressed; the additional lines will then be displayed etc., until the end of file is reached.

In the following example, the primary filename, DUMP distinguishes a category name of source files and the extension filename; ASM distinguishes the type of file for a particular category.

NOTE: The peripheral Interface Command (PIP) may also be used to display source files. The description of the PIP command is described under System Operation Commands.

```
Entry:                  0A> TYPE DUMP.ASM P23 cr
Syntax:                 0A> <TYPE> <DUMP> <.ASM> <P23> cr
```

where: TYPE = Command to display source file.
DUMP = Primary filename.
.ASM = Secondary file name extension.
P23 = Pause after 23 lines are displayed.
cr = Carriage return (advance to the next 23 lines and pause).

Command Name: FILE DIRECTORY

Function: The DIR (directory) command causes a list of the filenames that are stored in a diskette to be displayed at the terminal. The command can be invoked to list all filenames stored on the currently assigned disk drive, or all filenames stored on a specific disk drive, or list particular filenames stored on a specific disk drive. An error message is displayed if a requested filename cannot be found on the addressed disk drive.

The parameter field that is entered in the command string to identify the disk drive name (e.g., A: or B:) must be followed by a semicolon (;). If the drive name is not specified in the commands the system will address the last drive that was assigned.

To list all filenames on the currently assigned disk drive: Entering the command, DIR, followed by a <cr> will display all filenames on the currently assigned drive.

Entry: 0A> DIR cr
Syntax: 0A> <DIR> <cr>

System Response: All filenames on the logged-on disk drive will be displayed.

To list filenames stored on a specific disk drive: Entering the command, DIR, followed in sequence by the parameter for a specified disk drive, and a <cr> will cause the specified disk drive to be addressed and a search conducted to list all filenames stored on the diskette.

Entry: 0A> DIR B: cr
Syntax: 0A> <DIR> <B:> <cr>

where: Parameter B: indicates the specified drive.

System Response: All filenames on the specified drive will be displayed.

To list a particular filename on specified diskette: Entering the command, DIR, followed in sequence by the parameter for a specific disk drive, the two parameters for filename and/or extension, and a <cr> will cause the specified file to be displayed.

Entry 0A> DIR B: <*> .ASM cr
x: 0A> <DIR> <B:> <*> <.ASM> <cr>

where parameters: * = filename
.ASM = extension (file type)

Error Message: The message, NOT FOUND is displayed if the file is not stored on the specified disk.

Command Name: RENAME FILE

Function: The REN (rename) command allows the user to change the name of files stored on the disk. It is assumed that the currently assigned disk contains the old filename that is to be changed to a new filename.

To change existing filename, MYFILE.ASM to read YY.ZZ: Entering the REN command followed in sequence by (1) The existing filename and extension parameters, (2) An equal sign (=), (3) The new filename and extension parameters, and (4) A <cr> will initiate the change.

```
Example:        0A> REN MYFILE.ASM = Y.Y.ZZ cr
Syntax:         0A> REN <MYFILE> <.ASM> <=> <YY> <.ZZ> <cr>
```

where: The filename and extension parameters must be specified in the command string for both the current filename and new filename.

The optional disk drive name (e.g., A: or B:) may be used in the command string to identify the file location. If the current filename is assigned to a specified drive, then the new filename is expected to reside on the same drive. Likewise if the new filename is assigned to a specified drive, then the current filename is expected to reside on the same drive as indicated in the following examples:

```
Example Entry:  (1) 0A> REN B: MYFILE.ASM = B:YY.ZZ cr
                (2) 0A> REN B: MYFILE.ASM = YY.ZZ cr
```

where: Both examples (1) and (2) indicate the filename, MYFILE.ASM is changed to YY.ZZ on drive B.

```
(3) 0A> REN A: BAL.M = BAT.M cr
(4) 0A> REN A: BAL.M = A: BAT.M cr
```

where: Both examples (3) and (4) indicate the filename BAL.M is changed to BAT.M on drive A.

Command Name: STATUS

Function: The STAT (Status) command provides statistical information about file storage and device assignments. The STAT command is invoked as follows:

```
Entry:          0A> STAT cr
Syntax:         0A> <STAT> <cr>
```

Full details and variations for entering the STAT command are described in the MP/M and CP/M users manuals that are shipped with the system.

Command Name: GENMOD

aaneengeschakelde

Function: The GENMOD command accepts a file which contains two concatenated files of type HEX which are offset from each other by 0100H bytes, and produces a file of type PRL (program relocatable module). The GENMOD command is invoked in the following form:

```
Entry:  0A> GENMOD G: (filename) (.extension) B: (filename).PRL $1000
Syntax: 0A> <GENMOD> <B:> <file> <.hex> <B:> <file> <.PRL> <$1000>
```

where: The first parameter = filename which contains two concatenated files of type HEX extension.

The second parameter = the name of the destination file of type PRL extension.

The optional third parameter = the specification for additional memory space required by the program beyond the explicit code space. The form of this parameter is the dollar sign ($) followed by four hex ASCII digits. For example, if the the program has been written to use all available memory for buffers, the specificaiton of the third parameter will ensure that a minimum buffer allocation is provided.

Command Name: GENHEX

Function: The GENHEX command is used to produce a file of type HEX from a file of type COM. This capability allows the user to generate HEX files for GENMOD input. The GENHEX command has two parameters which consist of the COM filename and the foffset value fo the HEX file. The GENHEX command is invoked in the following form:

Entry: 0A> GENHEX PROG.COM 100
Syntax: 0A> GENHEX <PROG> <.COM> <100>

Command Name: PRLCOM

Function: The PRLCOM command accepts a file of type PRL (program relocatable module) and produces a file of type COM (memory resident). If the destination COM file already exists, a query is made to determine if the file should be deleted before continuing the command processing. The PRLCOM command is invoked in the following form:

Entry: 0A> PRLCOM B: (program) .PRL A: (program) .COM
Syntax: 0A> <PRLCOM> <B:> <program name> <PRL.> <A:> <program name> <.COM>

SYSTEM OPERATION COMMANDS

The system operation commands permit the user to perform the following functions:

- Transfer data files from one peripheral I/O device to another via the peripheral interchange program command.
- Assemble the users program and store the result on diskette.
- Select a file of commands for automatic batch processing via the submit command.
- Display the contents of a specified disk file on the screen in hexadecimal form via the dump command.

- Load a specified disk file with hexadecimal machine code and produce a memory image file which can be executed.
- Display run-time status of the MP/M Operating System via the system status command.
- Transfer ASCII text files to a list device via the spool command.

Command Name: PERIPHERAL INTERCHANGE PROGRAM

Function: The PIP (peripheral interchange program) command allows the user to initate data transfer operations between the disk file and other peripheral devices. The PIP command may be invoked by specifying the Interactive Response mode or Immediate Response mode. Variations for using the PIP are described in the CP/M Users Guide Manual.

To invoke PIP for Interactive Response: Entering the command PIP followed by a <cr> will call out the program

Entry:	0A> PIP cr
Syntax:	0A> <PIP> <cr>

The system will respond by displaying an asterisk (8) character which serves as a prompt for the user to begin entering one or more command lines at the console. The form of each command line entry is as follows:

* <destination> <=> <source #1> <,Source #2> ... <Source #n> <cr>

where: <destination> is the name of the file or peripheral device that will receive the specified source information.

<=> equal symbol is a delimiter

<source #1> through <source #n> represents a series of one or more files (or devices) that are copied from left to right and sent to the specified destination.

EXAMPLE COMMAND LINE ENTRY:

B: = *.COM cr -- Copy all files which have the secondary extension name, COM to drive B from the current drive.

Each command line entry causes some type of transfer function to occur as defined by the content in the command line.

The * prompt will continue to appear requesting input for a command line entry until a <cr> is issued immediately after the * prompt. This will cause the PIP to terminate.

To invoke PIP for Immediate Response: Entering the command PIP, followed in sequence by the command line entry and a <cr> will call out the program and execute the command without the need for user interaction. The program automatically terminates after execution is completed.

Entry: 0A> PIP (command line)
Syntax: 0A> <PIP> <destination> <=> <source #1> <source #2> ... <source #n> <cr>

where: description of parameters for command line entry is the same as given for interactive response.

Command Name: ASSEMBLER

Function: The ASM (assembler) command allows the user to assemble a specified program on a disk. The MP/M assembler is invoked as follows:

Entry: 0A> ASM (filename) (flags) cr
Syntax: 0A> <ASM> <filename> [F1 F2 F3] <cr>

where: The command ASM means assemble. The filename is the name of the source file to be assembled. The file type extension is not included in the command. MP/M uses the file name to generate:

- The source filename by appending .ASM
- The list filename by appending .PRN
- The object file name by appending .HEX

The three flags (F1, F2, F3) are optional. The first flag is associated with the source file, the second flag with the object file, the third flag with the list file. the flags are one character each and have the following meaning:

A through P - Logical disk drives

Z - Do not produce a file

X - Applies only to Flag 3 (F3) and means put the Listing on the console.

NOTE: If the filename has a logical drive designator (A:filename) and the flag specifies a different logical drive, the flag takes precedence.

Command Name: SUBMIT

Function: The SUBMIT command allows a user to combine several commands into a single file for automatic batch processing. The SUBMIT function creates a file of substituted commands with the name, $$$.SUB. The SUBMIT command is invoked as follows:

Entry: 0A> SUBMIT (filename) (parameter #1 ... parameter #n) cr
Syntax: 0A> <SUBMIT> <filename> <.extension> <parameter #1> <parameter #n> <cr>

where: Filename is the name of a file that exists on the currently assigned disk which contains an extension of .SUB. The complete filename is of the form, ASM.SUB.

A filename with the .SUB extension, type descriptor is therefore identified by the SUB file. The SUB file contains prototype commands with possible parameter substitution to accomplish batch processing.

The actual parameters #1 ... parameter #n entered in the command string represent commands contained in the prototype file. The prototype command file is created using the Editor program with interspersed ($) symbols for formal parameters of the form: $1 $2 $3 ... $n.

These formal parameters ($1 $2 $3 ... n$) correspond to the number of actual parameters (parameter #1 ... parameter #n) entered in the command string and are substituted into the prototype commands.

When the SUBMIT command is executed, the actual parameters (parameter #1 ... parameter #n) are paired with the formal parameters ($1 $2 $3) in the prototype commands.

If the number of formal and actual parameters correspond, batch processing is initiated and the file of substituted commands are executed in sequence.

If the number of formal and actual parameters does not correspond, then the SUBMIT function is aborted with an error message displayed at the console.

Command processing is also aborted if the system detects an error in any of the commands. The user can abort command processing at any time by typing a rubout when the command is read or echoed.

Example:

Let us assume the file ASMBL.SUB exists on disk and contains the following prototype commands

```
ASM $1
DIR $1.*
ERA *.BAK
PIP $2:=$1.PRN
ERA $1.PRN
```

and the SUBMIT command is issued by the operator:

```
SUBMIT ASMBL X PRN cr
```

The SUBMIT program reads the ASMBL.SUB file, substituting "X" for all occurrences of $1 and "PRN" for all occurrences of $2, resulting in a $$$.SUB file containing the following commands which are executed in sequence.

```
ASM X
DIR X.*
ERA *.BAK
PIP PRN:=X.PRN
ERA X.PRN
```

The SUBMIT function can also access a SUB file which is stored on an alternate drive by preceding the filename by a drive name. Submitted files are nly acted upon, however, when they appear on drive A. Thus, it is possible to create a submitted file on a drive B diskette which is executed at a later time when the diskette is inserted in drive A.

Command Name: DUMP

Function: The DUMP command types the contents of the specified disk file at the console in hexadecimal form. The contents are listed 16-bytes as a time, with the absolute byte address listed in hexadecimal to the left of each line. The DUMP command is invoked as follows:

```
Entry:      0A> DUMP (filename) cr
Synatax:    0A> <DUMP> <filename> <.extension> <cr>
```

where: Filename and extension is of the form: BOT.ASM

Long listings can be aborted by pressing the rubout key while printing is in process.

Command Name: LOAD

Function: The LOAD command reads the specified filename, which is assumed to contain hexadecimal format machine code and produces a memory-image file that can be subsequently executed. The LOAD command is invoked as follows:

```
Entry:      0A> LOAD (filename) cr
Syntax:     0A> <LOAD> <filename> <.extension> <cr>
```

where: Filename and extension is of the form: ALPA.HEX

Command Name: SYSTEM STATUS

Function: The MPMSTAT (MP/M System Status) command allows the user to display the run-time status of the MP/M operating system. The MPMSTAT command is invoked as follows:

```
Entry:      0A> MPMSTAT cr
Syntax:     0A> <MPMSTAT> <cr>
```

The system status is displayed on the screen in the following format:

```
****** MP/M Status Display ******

Top of memory = FFFFH
Number of consoles = 02
Debugger breakpoint restart # = 06
Stack is swapped on BDOS calls
Z80 complementary registers managed by dispatcher
Ready Process(es):
  MPMSTAT Idle
Process(es) DQing:
  [Sched   ] Sched
  [ATTACH  ] ATTACH
  [CliQ    ] cli
Process(es) NQing:
Delayed Process(es):
Polling Process(es):
  PIP
Process(es) Flag Waiting:
  01 - Tick
  02 - Clock
Flag(s) Set:
  03
Queue(s):
  MPMSTAT  Sched       CliQ     ATTACH    MXParse
  MXList   [Tmp0   ]   MXDisk
Process(es) Attached to Consoles:
  [0] - MPMSTAT
  [1] - TMP1
Memory Allocation:
  Base = 0000H  Size = 4000H  Allocated to PIP     [1]
  Base = 4000H  Size = 2000H  * Free *
  Base = 6000H  Size = 1100H  Allocated to DIR     [0]
```

The information presented in the status display is to be interpreted as follows:

Ready Process(es): The ready processes are those processes which are ready to run and are waiting for the CPU. The list of ready processes is ordered by the priority of the processes and includes the console number at which the process was initiated. The highest priority ready process is the running process.

Process(es) DQing: The processes DQing are those processes which are waiting for messages to be written to the specified queue. The queue name is in brackets followed by the names of processes, in priority order, which have executed read queue operations on the queue.

Process(es) NQing: The processes NQing are those processes which are waiting for an available buffer to write a message to the specified queue. the queue name is in brackets followed by the names of the processes, in priority order, which are waiting for buffers.

Delayed Process(es): The delayed processes are those which are delaying for a specified number of ticks of the system time unit.

Polling Process(es): The polling processes are those which are polling a specified I/O device for a device ready status.

Process(es) Flag Waiting: The processes flag waiting are listed by flag number and process name.

Flag(s) Set: The flags which are set are displayed.

Queue(s): All the queues in the system are listed by queue name. Queue names which are all in capital letters are accessible by command line interpreter input. For example, the SPOOL queue can be sent a message to spool a file by entering 'SPOOL' followed by a filename. Processes DQing from queues which have a name that matches the process name are given the console resource when they receive a message. Queue names that begin with 'MX' are called mutual exclusion queues. The display of a mutual exclusion queue includes the name of the process, if any, which has the mutual exclusion message.

Process(es) Attached to Consoles:: The process attached to each console is listed by console number and process name.

Process(es) Waiting for Consoles: The processes waiting for each console are listed by console number and process name in priority order. They are processes which have detached from the console and are then waiting for the console before they can continue execution.

Memory Allocation: The memory allocation map shows the base, size, bank, and allocation of each memory segment. Segments which are not allocated are shown as '* Free *', while allocated segments are identified by process name and the console in brackets associated with the process. Memory segments which are set as pre-allocated during system generation by specifying an attribute of 0FFH are shown as '* Reserved *'.

Command Name: SPOOLER

Function: The SPOOL command allows the user to transfer (spool) ASCII text files to the list device. Multiple filenames may be specified in the command tail. The spooler expands tabs (ctl-I characters), assuming tab positions are set at every eighth column.

The spooler queue can be purged at any time by using the STOPSPLR command.

The SPOOL command is invoked as follows:

```
Entry:  0A> SPOOL (filename), (filename) cr
Syntax: 0A> <SPOOL> <LOAD> <.LST> <,> <LETTER> <.PRN> <cr>

where:  SPOOL          = the command entry
        LOAD.LST       = the first filename with extension
        Character (,)  = a delimiter to separate filenames
        LETTER.PRN     = the second filename in command string
```

The non-resident version of the spooler (SPOOL.PRL) differs in its operation from the SPOOL.RSP as follows: it uses all of the memory available in the memory segment in which it is running for buffer space; it displays a message indicating its status and then detaches from the console; it may be aborted from a console other than the initiator only by specifying the console number of the initiator as a parameter of the STOPSPLR command, which is invoked as follows:

```
Entry:                  0A> STOPSPLR 0 cr
Syntax:                 0A> <STOPSPLR> <0> <cr>

where:  STOPSPLR = the command entry
                0 = the console # of intiator
```

PROGRAM OPERATION COMMANDS

The program operation commands allow the user to perform the following functions:

- o Invoke the MP/M Text Editor Utility
- o Invoke the Dynamic Debugger Utility
- o Examine and Set the System Date and Time Parameter
- o Schedule Programs for Execution
- o Abort a Running Program

Command Name: TEXT EDITOR

Function: The ED (editor) command allows the user to create and edit ASCII text files using the MP/M editor.

The alternate text editor program that is described in chapter 6 may be used to provide an extended ASCII text processing capability for 9520 Software Development System applications.

Complete details for using the MP/M ED utility are described in the CP/M Editor Users Manual that is shipped with the system.

The MP/M ED command is invoked as follows:

```
Entry:                  0A> ED cr
Syntax:                 0A> <ED> <cr>
```

Command Name: DYNAMIC DEBUGGING TOOL

Function: The DDT (dynamic debugging tool) command loads and executes the MP/M debugger.

The description for using the DDT is presented in chapter 6 of this manual. Additional information on the DDT is conatined in the MP/M Users Manual that is shipped with the system.

The DDT command is invoked as follows:

Entry:	0A> DDT cr
Syntax:	0A> <DDT> <cr>

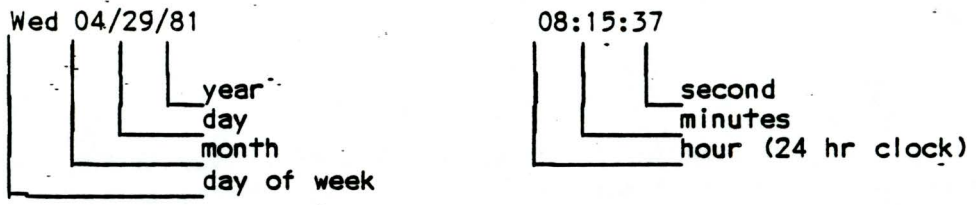
Command Name: DATE AND TIME

Function: The TOD (time of day) command allows the user to read and set the date and time. Entering 'TOD' followed by a <cr> will cause the current date and time to be displayed on the console. Entering 'TOD' followed by a date and time will set the date and time when a <cr> is entered following the prompt to strike a key. Each of these TOD commands is illustrated below:

To Display the Current Date and Time:

Entry:	0A> cr
Syntax:	0A> <TOD> <cr>

System Response: The screen will display the following information:



To Set the Date and Time:

1) Entry: 0A> <TOD> (date) (time)
Syntax: 0A> <TOD> <04/29/91> <10:30:00> <cr>

2) System Response message: STIKE KEY TO SET TIME

3) User Response: press <cr> key

4) System Response: New Time and Date is displayed as follows:

WED 04/29/81 10:30:00

NOTE: Entering TOD P will cause the current time and date to be displayed continuously until a key is struck at the console.

Command Name: SCHEDULER

Function: The SCHED (scheduler) command allows the user to schedule a program for execution. Entering 'SCHED' followed by a date, time and command line will cause the command line to be executed when the specified date and time is reached.

In the example shown below, the program 'SAMPLE' will be loaded from disk and executed on April 18, 1981 at 10:30 PM. Note that only hours and minutes are specified, not seconds. Programs are scheduled to the nearest minute.

The command is invoked as follows:

Entry:	0A> SCHED (date) (time) (program name) cr
Syntax:	0A> <SCHED> <4/18/81> <22:30> <SAMPLE> <cr>

Command Name: ABORT

Function: The ABORT command allows the user to abort a running program. The program to be aborted is entered as a parameter in the ABORT command.

The command in invoked as follows:

Entry:	0A> ABORT (program name) cr
Syntax:	0A> <ABORT> <RDT> <cr>

A program that is initiated from another console may only be aborted by including its console number as a parameter of the ABORT command. The console entry is as follows:

Entry:	0A> ABORT (program name) (console #) cr
Syntax:	0A> <ABORT> <RDT> <2> <cr>

SYSTEM UTILITIES

System Utilities is a collective name for programs which provide various system overhead operation such as formatting diskettes, setting baud rates, converting absolute object files to a suitable format for downloading to a remote emulator/debugger, downloading a program from the 9520 Software Development System to the 9508 Emulator/Debugger and uploading a program from the 9508 Emulator/Debugger to the 9520 Software Development System.

The following utility programs are used to accomplish the above operations:

- FDISK (Floppy Disk)
- BRATE (Baud Rate)
- CONVERT
- DOWNLOAD
- UPLOAD

There are two types of utilities command entries: immediate (single entry) commands, and interactive (user response) commands. The Immediate and Interactive modes of command entries are described in chapter 8, System Operations.

Utility Name: FLOPPY DISK

Function: The FDISK (floppy disk) ulility allows a user to perform the following functions:

- o FORMAT a floppy diskette
- o Duplicate a floppy diskette
- o Copy the system tracks from one diskette onto another diskette

The FDISK utility may be invoked using the interactive mode which prompts the user for a response, or using the immediate mode which initiates the execution and completion of the program.

To invoke the FDISK utility using the interactive mode:

Entry: 0A> FDISK cr
Syntax: 0A> <FDISK> <cr>

System Response: FDISK will prompt the user in the following manner:

F) ormat? D) uplicate? C) opy System? Q) uit?

The user should respond by typing the character F for formatting the disk, D for duplicating the disk, C for copying the system tracks, and Q for aborting the FDISK program.

Upon reception of a valid character, FDISK will prompt the user according to the response entered. For the F character response, FDISK will prompt with:

Diskette to format - A, B, C, or D?

The user may choose the diskette to format by striking the letter A for the diskette in drive A, B for the diskette in drive B, and so on...

FDISK will then respond with:

D) ouble Density? S) ingle Density?

The user may choose double density with the D character or single density with S character. Upon receiving either one of these characters, FDISK will format the appropriate floppy diskette and then respond with the first prompt again.

When the D chartacter is chosen from the first prompt issued by FDISK, the utility will again prompt with:

Source diskette - A, B, C or D?

The user may choose the diskette to be duplicated by striking the letter A for the diskette in drive A or B for the diskette in drive B and so on . . .

FDISK will then prompt with:

Destination diskette - A, B, C, or D?

The user may choose the new disk that will receive the duplicated data by striking the letter A for the diskette in drive A, B for the diskette in drive B, and so on . . .

FDISK will then duplicate the destination diskette with the kcontents of the source diskette and then respond with the first prompt again.

When the C character is chosen from the first prompt issued by FDISK, the utility will perform the same action as that taken for the duplicate function except that only the first two tracks of the source diskette will be updlicated onto the first two tracks of the destination diskette.

To invoke the FDISK utility using the immediate command mode:

Entry: 0A> FDISK (function) (flag #1) (flag #2) cr
Syntax: 0A> <FDISK> <function> <flag #1> <flag #2> <cr>

where: Function means:

FORMAT
DUPE
CPYSYS

and FLAG #1 means:

DRIVE = for the FORMAT indicator.
FROM = for the DUPE and CPYSYS indicators.

and FLAG #2 means:

DENSE = for the FORMAT indicator.
TO = for the DUPE and CPYSYS indicators.

These parameters to FDISK will cause the utility to execute without prompting the user. Also, these parameters may appear on the command line in any order or sequence. That is to say:

There may be 6 variations of the command line for the FORMAT indicator. For example:

FDISK	FORMAT	DRIVE = B	DENSE = D
FDISK	FORMAT	DENSE = D	DRIVE = B
FDISK	DENSE = D	FORMAT	DRIVE = B
FDISK	DENSE = D	DRIVE = B	FORMAT
FDISK	DRIVE = B	FORMAT	DENSE = D

Of course A, B, C or D characters may follow the equal symbol (=) for the DRIVE, FROM, and TO indicators and S, or D characters may follow the = sign for the DENSE indicator.

Utility Name: BAUD RATE

Function: The BRATE (baud rate) utility program will allow the user to display and set baud rates for the 9520 Development System I/O devices. These devices include RS-422 I/O, Remote I/O, Reader/Punch I/O and Console I/O communication ports.

The BRATE utility may be inviked using the interactive mode which prompts the user for a response, or by using the immediate mode which initiates the execution and completion of the program.

To invoke the BRATE utility using the interactive mode:

Entry: 0A> BRATE cr
Syntax: 0A> <BRATE> <cr>

System Response: The BRATE program will then display the current baud rate status for each I/O device and then prompt the user as follows:

I/O device:

The user may respond with the ASCII description of the I/O device and enter a <cr> after each response:

RS-449 <cr>
REMOTE <cr>
PRINTER <cr>
CONSOLE <cr>

The baud rate program will then prompt the user for the baud rate that is desired:

BAUD RATE:

The user may respond by entering the following values for the baud rate.

where: Valid Baud Rate Values are:

110
134.5
150
300
600
1200
2400
4800
9600
19200
38400
56000
76800
187500
375000
750000

NOTE: Error messages will be displayed for specifying those baud rates that are invalid for a particular I/O device.

A control - C character may be entered to abort thge baud rate program at any time.

To invoke the BRATE Utility using the immediate mode: All information is provided in the parameters of the command string and immediate execution occurs to complete the processing.

Entry: 0A> BRATE (I/O device) = (Baud Rate) cr
Syntax: 0A> <BRATE> <I/O device> = <Baud Rate> <cr>

where: I/O device is the ASCII description for one of the following I/O ports:

RS-422
REMOTE
PRINTER
CONSOLE

The equal symbol (=) is a delimiter

Baud Rate is one of the following values shown in the preceding table of "Valid Baud Rate Values".

Utility Name: CONVERT

Function: The CONVERT utility converts an absolute object file, which has been produced by the Millennium Assembler or Linker, to a file which is suitable for downloading to a 9508 for debugging, or to a PROM programmer. The format of the output file can be either Millennium Binary or TEKHEX. Note that PROM Programmers only support the TEKHEX format.

The CONVERT utility is invoked as follows:

Entry: 0A> CONVERT (d:) (fn) (.ft) (/A) cr
Syntax: 0A> <CONVERT> <drive:> <filename> <.file type extension> </A > <cr>

where:

drive: Is the disk drive letter. A, B, C, or D. The default is the drive which is currently logged on.

filename: Is the filename.

filetype: Is the filetype extension of the object file. The default is OBJ.

/A Instructs the CONVERT program to create TEKHEX records.

In this case, the file type of the output file will be .ASC. The default format is Millennium Binary, and the corresponding file type is .BIN.

System Response: If no errors are encountered, the CONVERT will display

FILE CONVERTED. NO ERRORS

and return to the operating system.

Warning Messages:

If the input file contains records which have relocation information the CONVERT program will display a warning message and continue with its conversion. The message will be one of the following:

INPUT MODULE HAS UNRESOLVED EXTERNAL REFERENCE
INPUT MODULE IS NOT ABSOLUTE. IT CONTAINS RELOCATION INFO

Fatal Errors:

If the switch parameter, /A, is not typed correctly, the program will display

INVALID PARAMETER. EXPECTING A SWITCH

If an error occurs while reading the input file or writing the output file, the error message will consist of two lines. The first line will always be:

FATAL ERROR nn fn.ft. RECORD = rr

where: nn is the error number

fn is the filename

ft is the file type. For input errors, it will be the file type of the input file. For output errors, it will be the file type of the output file.

rr is the physical record number in the File Control Block (FCB)

The second message and their corresponding error codes are

CODE	MESSAGE
66	CHECKSUM DOES NOT MATCH
77	INPUT FILE HAS A SHORT BLOCK
xx	OUTPUT ERROR
xx	CANNOT OPEN FILE
xx	CANNOT CLOSE FILE

xx is the code which is returned by MP/M for an OPEN, ERASE, MAKE, READ, WRITE, or CLOSE BDOS call.

Utility Name: DOWNLOAD

Function: This utility program downloads a program from a file on the 9520 Development System to the 9508 Emulator/Debug system so that it can be debugged. The input file must be in Millennium Binary or TEKHEX format. The link must be connected to a CONSOLE PORT on the 9520.

The DOWNLOAD utility is invoked in the following manner:

Entry: 0A> DOWNLOAD (d:) (fn) (.ft) cr
Syntax: 0A> <DOWNLOAD> <drive:> <filename> <.file type extension> <cr>

where:

drive: is the disk drive letter. A, B, C, or D. The default is the current logged on drive.

filename: is the filename of the input file.

file type: is the file type extension. The default is BIN.

NOTES: This program uses:
1) Start-Synchronization Handshake
2) Ack/Nak Protocol

Utility Name: UPLOAD

Function: The UPLOAD utility is used to upload a program from the 9508 Emulator/Debug System to a disk file on the 9520 Development System. It assumes that the link from the 9508 is connected to a CONSOLE PORT.

The UPLOAD utility is invoked in the following manner:

Entry: 0A> UPLOAD (d:) (fn) (.ft) cr
Syntax: 0A> <UPLOAD> <drive:> <filename> <.file type extension> <cr>

where:

drive: is the disk drive letter. A, B, C, or D. The default is the current logged-on disk.

filename: is the filename of the output file.

file type: is the file type extension. Normally BIN for Binary format and ASC for TEKHEX format. The default is BIN.

NOTES: This program uses:
1) Start-Synchronization Handshake
2) Ack/Nak Protocol

THE ASSEMBLER

The Assembler is a 9520 Development System utility that assembles assembly language source programs into object modules for execution (after suitable link operation and downloading has been completed) at the remote 9508 Emulator/Debug System.

A separate assembler utility is used for each microprocessor type. In some cases, one assembler will be used to support a family of processor types. Assembler directives in the source program will be used to specify the particular microprocessor family member. The details of the assembler directives can be found in the assembler manuals. A separate user manual and language translator diskette is provided for each assembler/cross-assembler to suppor the various microprocessors. The available assembler manuals are listed in the preface.

The assembler is invoked in the following manner:

Entry: (assembler specifier) (filename) (.file type) (>flags) cr
Syntax: <assembler specifier> <filename> <.file type> (>flags) <cr>

where parameters in the command entry indicates the following information:

(Assembler Specifier Parameter) is defined by the following entry

A8080 - The 8080 and 8085 assembler

A6800 - The 6800 assembler

A6801 - The 6801, 6802, and 6803 assembler

A6809 - The 6809 assembler (future capability)

A8048 - The 8048, 8049, 8021, and 8041 assembler (planned)

AZ80 - The Z80 assembler

AZ8001 - The Z8001 assembler (future capability)

AZ8002 - The Z8002 assembler

A8086 - The 8086 and 8088 assembler (future capability)

A68000 - The 68000 assembler (future capability)

(filename parameter) is used by the command processor to specify the primary name of the source file (.SRC), the object fiel (.OBJ), the list file (.LST) by appending the appropriate file type name.

Notice that the file type name selected are different than the MP/M file type names. This is necessary to avoid mixing the Millennium object files with those of Digital Research. The object file formats are not compatible. A blank must separate all fields in the command.

NOTE: The Symbol Table file will not be generated in the first release . 0 (>flags) is an optional field that is used to specify flags to be applied to the source, list, object and symbol table files in the foregoing order. The flags are the same ones used in the MP/M ASM command.

A, B, C, D - Logical drive designators

Z - no file is to be produced

T - Applies only to the list file and means use the system printer instead of a file

X - Applied only to the list file and means use the console instead of a file

The assembly command allows one source file to be specified. At times the user may want to concatenate several source files and submit them as one source program. This is accomplished by building a file that has INCLUDE statements. For example, assume the user has three source files (B:FILEA.SRC, A:FILEB.SRC, FILEC.SRC) that contains the following statements:

```
ORG 100H
INCLUDE B:FILEA.SRC
INCLUDE A:FILE.SRC
INCLUDE FILEC.SRC
```

In the assemble command, the name SOURCE.SRC is given for the source filename. The assembler will open the file SOURCE.SRC, find the first INCLUDE statement, then open B:FILEA.SRC and process it, find the second INCLUDE, then process A:FILEB.SRC. The third INCLUDE statement specifies FILEC.SRC. The default Logical drive designator A will be prefixed, making it A:FILEC.SRC.

NOTE: Include files can not be nested; that is one include file cannot call another include file.

THE LINKER

The linker is a 9520 Development System utility program. It produces an executable load module by linking relocatable object modules produced by the 9520 assemblers and compilers. The Linker input therefore comes from the assembler.

The object modules output from the Assembler consists of Text Blocks, Relocation Blocks, and Global Symbol Directory Blocks. Text Blocks from an independently assembled program section consist of three types of information.

1. Constants and machine instructions whose values are independent of their position in memory;

2. Addresses or address constants whose values are relative to the starting location (base) of a section; and

3. Global references to other object modules whose values cannot be determined until all sections are assigned memory locations.

Relocation Blocks contain information necessary to update and relocate bytes of program text. Global Symbol Directory Blocks define global symbols and sections.

The Linker supports the unique qualitites of each of the microprocessors supported by the 9520 Development System. The Linker's outward appearance and its operational method remain the same, regardless which microprocessor is supported.

To prepare object modules for the CONVERT utility command, the Linker performs three specific functions for each module, in the order of entry:

1. Allocates memory space for each section of the load file;

2. Establishes a reference table of global symbols; and

3. When necessary, relocates address-dependent locations to correspond to allocated space.

In addition, the Linker generates a listing that indicates where sections are allocated and states the values of all global symbols.

Invoking the Linker

The linker may be invoked by one of three available methods:

- Invoke Simple Linker
- Invoke Linker Interactive Command
- Invoke Linker Command File

The method used to Invoke Simple Linker requires the entry of filenames only. All other parameters are set to reasonable default values. This method is usually adequate for most linking situations.

The Interactive Command method for invoking the linker causes the program to issue prompts to the user for supplying the required input.

The Command File method for invoking the Linker requires the user to specify a filename that contains a linker command series designator.

Essentials for Entering Linker Commands

Extensive use is made of title names in the linker commands. A brief summary of filename conventions will be given here to facilitate describing the Linker commands. A filename has three parts:

1. Logical drive designator - The first character, followed by a colon (:) specifies a logical drive. The letters A, B, C, and D specify logical drives. If a logical drive is not specified the system defaults to logical drive A.

2. Filename - This specifies the actual primary name of the file. A filename has a maximum of eight (8) characters. (See the MP/M Manual for details.)

3. File type - File type is the secondary name extension of a file and is preceded by a period (.). File type is specified with three characters. The file types that are of interest with regard to the linker are:

   a. OBJ - Means object file. Object files are produced by the assemblers and the Linker.

   b. LST - Means a list file. These are files that will be listed on the printer.

Invoke Simple Linker

The simple form of invoking the LINK command is:

```
Entry:  0A> LINK (filename) = S1 S2 > flags cr
Syntax: 0A> <LINK> <filename> = <S1> <S2> (> FLAGS) <cr>
```

where:

1. LINK invokes the Linker

2. Filename -- specifies the names of the two (2) files to be produced.

   where:

   The Load File name is produced by appending .OBJ

   The Load File name is produced by appending .LST

3. S1, S2 -- Specifies the filenames of the relocatable files that are to be used as input.

4. > FLAGS -- Specifies flags that are to be applied to the Load File name and the List File, respectively.

   The flags are:

   a) A through P - Logical drive designators

   b) Z - No file is to be produced

With simple invocation, all of the command must appear on one line.

Invoke Interactive Command

The Interactive Command Link is invoked as follows:

```
Entry:                  0A> LINK cr
Syntax:                 0A> <LINK> <cr>
```

System Response: The Linker responds with a prompt character (*), to indicate that Linker commands will be accepted. Each command is terminated with a carriage return. Commands are accepted until the END command is received. The END command directs the Linker to discontinue command entry and to begin processing the object files. See Linker Commands for a list of legal commands.

Invoke Linker Command File

The Linker Command File is invoked as follows:

Entry: 0A> LINK @ filename cr
Syntax: 0A> <LINK> <@> <filename> <cr>

System Response: The Linker opens the file specified by (filename) and reads commands from the file until an end-of-file or and END command is encountered. End-of-file or END directs the Linker to begin processing the object modules. If errors have been generated, the Linker aborts with the message:

ERROR IN INDIRECT FILE, LINK ABORTED

See Linker Commands for a list of legal commands.

The following commands may be used in invoking the interactive or command file modes:

LOG

Print messages to the console and log commands on the list file, if one has been specified. All commands are echoed to the Linker list file after LOG has been indicated.

NOLOG

Do not log Linker messages on the console.

MAP

Generate a memory map in the linker list file. A memory map lists module names, section names and attributers, entry points within sections, and undefined global symbols. (See the Linker Output description in this section.)

NOMAP

Do not generate a memory map.

LIST (filename)
(device:)

Generate a Linker List file named filename. See the listing file description for contents of the Linker list file. "Filename" is any valid file specification. Instead of a filename, the printer or console may be designated with a (T:) or (X:), respectively.

LOAD (filename)

Generate a load file named "filename". The file will contain the executable output of the Linker and can be downloaded using the DLOAD command.

DEFINE (symbol1 = value) (,symbol2 = value) . . .

Define symbols, symbol1 and symbol2 ... are names of global symbols. Value is a hexadecimal number.

LINK filename, filename, filename, - - Link object files. This command directs the Linker to include the specified object modules in the download file.

```
                                                                       ,PAGE
LOCATE          [ , BASE (starting address)                      ]   [ ,INPAGE ]
(section name)  [ , RANGE (starting address, ending address)     ]   [ ,BYTE   ]
```

Locate a section and/or redefine its relocation type. Note that redefining the relocation type of a section may cause the linked code to execute differently than intended.

where:

section name - is the name of the section to be allocated.

BASE - is the hexadecimal starting address.

RANGE - is the hexadecimal starting and ending addresses. If there is not enough space within the specified range, the section will not be linked.

PAGE - is relocation type causing relocation on any page boundary. The size of a page is microprocessor dependent.

INPAGE - is a relocation type; causing relocation on any byte address, provided the section does not extend across page boundaries.

BYTE - is a relocation type; causing relocation at any byte address.

@filename

Indicates indirect command file. This command directs the Linker to obtain subsequent commands from filename. Commands are read from filename until an end of file or an END command is encountered. Indirect commands are echoed on the console as they are read, if LOG is specified. Nested indirect command files are illegal; a command file may not contain the @filename command.

TRANSFER (symbol)
(value)

Specify load module transfer address. Symbol is a global symbol and value is a hexadecimal number with a leading character ranging from 0 through 9. This transfer value supersedes any transfer address encountered in linking object modules.

END

End command entry mode. If no errors have been generated in command file invocation, this command will terminate command entry mode and initiate the processing of object modules. If errors are detected, an appropriate message is issued and control is returned to the system console.

COMMAND PROCESSING ERRORS

Extraneous Information Ignored

Extra characters are on a command line that only requires an instruction (e.g., LOG, NOLOG, MAP). The Linker performs the appropriate action for the command, ignoring extra characters on the line.

Illegal Command

The command was not recognized.

Syntax Error

Statement syntax is invalid. This error occurs when a command is incorrectly formed. For example, unmatched parentheses are found in the LOCATE command, or an operand is missing after the equals sign in the DEFINE command.

Indirect File Depth Exceeded

A filename command was found during processing of an indirect command file. The command is ignored.

Invalid Filename

The file in a LIST, ULOAD, or LINK command contains illegal file characters. (See the MP/M Manual for filename details.)

NOTE: Processing of the command line ceases when an invalid filename is encountered. All files up to the invalid filename, in the case of the LINK command, are added to the list of files to be linked.

Invalid Range Specified

The range (starting address through ending address) in the LOCATE command is invalid. The ending address must be greater than the starting address.

LINKER EXECUTION

Program Sections

A section is a collection of object code that has been assembled with the same location counter. An object code module may consist of several sections. These sections are treated separately by the Linker and each seciton is independently relocatable. No limit is placed on the number of sections per link, but no more than 255 sections or globals may exist in any one object module.

A section has five attributes that provide the Linker with information regarding memory allocation and where to link the section. These attributes are name, section type, size, relocation type, and memory location.

NAME

A section has a name consisting of up to eight characters, assigned by the sectin directives, SECTION, RESERVE, or COMMON at assembly time. The name must be a valid identifier. The section name is entered into the Linker's symbol table and is a valid external symbol.

SECTION TYPE

A section may be either a Section, Reserve, or Common. The specification is made through use of the SECTION, RESERVE, or COMMON directive at assembly time.

Each Section name must be unique. Multiple Sections with the same name will be flagged as errors, and only the first one will be linked.

Reserve sections with the same name are concatenated by the Linker. The length of a Reserve sectin in a load module is the sum of all Reserve sections with the same name.

Common sections with the same name are allocated the same space in memory. The length of the linked Common is that of the largest Common section.

SIZE

The size of each section in an object module is determined at assembly time. Section size is the number of program memory bytes that the section may occupy.

RELOCATION TYPE

A section may be absolute (non-relocatable), byte-relocatable, page-boundar-relocatable, or inpage-relocatable.

An absolute section is not reloacted by the Linker. Memory locations in an absolute section where code has been generated, or where locations have been explicitly reserved by the Assembler BLOCK directive, are not allocated to any relocatable section at link time. However, if two or more absolute sections have code at the same address, the contents of those memory locations after linking are undefined. These memory conflicts, if they occur, are noted on the Linker memory map.

A byte-relocatable section can be placed anywhere in memory.

MEMORY LOCATION

At link time the user may specify a relocatable section location, in the form of either a base address or an address range where the section may be placed. The default range for a relocatable section is the entire address space of the microprocessor. If the user elects not to specify a location for a section, the Linker will locate the section. An absolute section cannot be moved at link time.

The Default Section

If no SECTION directive is entered before assembly, the entire module is considered to be a byte-relocatable section with the same name as the object module.

Memory Allocation of Sections

The Linker allocates memory in the following sequence:

1. Absolute sections.
2. Based sections. Based means a program section starting location has been specified by a LOCATE command.
3. Ranged page-relocatable section*. Ranged means the user has explicity declared a RANGE (starting address, ending address) with the LOCATE command at link time.
4. Ranged inpage-relocatable sections*.
5. Ranged byte-relocatable sections*.
6. Page boundary-relocatable section*.
7. Inpage-relocatable section.
8. Byte-relocatable sections.

*Range was declared at link time.

Absolute and based sections are linked even if conflicts occur. A conflict exists when two or more sections have bytes at the same address. Other section types are not linked if a conflict occurs. If any memory conflict occurs during allocation, the conflict is noted on the memory map. The content of memory in the conflicting area is undefined.

ENDREL

ENDREL is a pre-defined symbol whose value is assigned at link time. After memory is allocated, ENDREL is assigned the value of the first memory address available for use. This address is one greater than the highest address used by a non-based relocatable section. All relocatable secitons are located below the value of ENDREL. Absolute sections, or sections relocated using the LOCATE command with a BASE specified, may or may not be located above the ENDREL address.

The user can override the value of ENDREL by assigning any other value to ENDREL. If ENDREL is neither defined nor referenced, no value is assigned.

LINKER OUTPUT

Linker Listing File

The listing file may be output either to a flexible disc file or to the console, line printer, or other output device.

The following information may be included in a Linker output listing:

Output Content Listed	Command File Invoked	Simple Linker Invoked
Global Symbol List	Yes	Yes
Internal Symbol List	If specified	Yes
Map	If specified	Yes
Linker Statistics	Yes	Yes
Error Message	If specified	Yes

Global Symbol List

A global symbol list is an alphabetical list of all global symbols (sections and symbols) and their assigned values. If a symbol is undefined, its value field contains asterisks.

Internal Symbol List

The internal symbol list contains all symbols in the source file and their actual values. The list consists of three parts:

1. Scalars.
2. Alphabetical list of labels for each section.
3. Alphabetical list of labels for each unbound global.

If there are no labels for a section or global, then no list for that section or global is output.

The internal symbol list will be displayed only if the DBG parameter was entered with the LIST directive before assembly.

Linker Statistics

The Linker Statistics include the number of errors, the number of undefined symbols, the number of sections, the number of modules, and the transfer address.

```
1 ERROR                    1 UNDEFINED SYMBOL
3 MODULES                  6 SECTIONS
TRANSFER ADDRESS IS 0040
```

The TRANSFER ADDRESS identifies program starting location.

Error Messages

Three classes of errors can be generated during Linker execution.

WARNING (W)

A problem may exist but the linked program can probably be executed.

ERRORS (E)

Linked program probably will not execute properly.

FATAL ERRORS (F)

Error directly affecting the Linker's execution. The Linker closes all files and returns control to MP/M.

All errors cause message to be output to the LOG and LIST file or device. A fatal error will be output to the console even if NOLOG was specified.

In the following list, each error message is indicated as being Warning (W), an Error (E), or a Fatal Error (F).

F. LINKER INTERNAL ERROR AT nnnn

An error occurred in the Linker. Try linking again. If this error persists, carefully dodument the incident and submit an Software Performance Report to Millennium.

E. NO ROOM IN RANGE nnnn-nnnn FOR SECTION name

The section length is greater than available contiguous memory in range nnnn-nnnn of allocated section memory.

W. SECTION name CHANGED FROM INPAGE TO (BYTE) RELOCATABLE
(PAGE)

Section length is greater than the page size of the microprocessor. This could occur if several inpage reserve secitons were linked together and their total size exceeded the page size of the microprocessor. A section declared to be inpage relocatable, in a LOCATE command, will generate this error if the section exceeds microprocessor page size. If section size exceeds available page size, relocation will then be to a byte boundary.

F. INVALID OBJECT CODE FORMAT FOR FILE name
LOCATION = nnnn

The information in file is not valid input object format. Make certain that all files to be linked have been assembled. Location is the internal Linker address where the object file error was detected.

F. UNABLE TO ASSIGN file or device name

A filename specified as an input object module does not exist, or file/device is unavailable.

F. MEMORY FULL

Linker memory is totally allocated and linking has been terminated. The total number of globals, sections, or object modules must be reduced in order to link in the available memory.

W. TRANSFER ADDRESS UNDEFINED

No transfer address was specified to the Linker either through the TRANSFER command or by specifying "END (expression)" during assembly. When no transfer address is specified, the Linker creates transfer address 0.

W. TRANSFER ADDRESS MULTIPLY DEFINED IN MODULE name FILEname

The module has attempted to redefine the transfer address previously specified by a linked module or by the TRANSFER command. The Linker uses the first encountered transfer address to generate a transfer address for the load module. If no transfer address is specified, a transfer address of 0 is generated.

W. RELOCATION TYPE OF SECTION name MULTIPLY DEFINED IN MODULE name FILEname

An attempt was made to redefine the section relocation type (byte, page, inpage, or absolute). This occurs when the LOCATE command defined a relocation type differing from that specified at assembly time. The error also occurs when relocation attributes of a COMMON or RESERVE section differ between modules. The Linker uses the first encountered relocation attribute to define the section.

E. Symbol name MULTIPLY DEFINED IN MODULE name FILEname

Indicates that an attempt was made to redefine a global symbol or section. This error occurs when two modules both define a global of the same name or when two sections have the same name. Code section names must be unique. In the event of multiply defined sections, the Linker will only include the first one in the load file.

W. TRUNCATION ERROR AT nnnn IN MODULE name FILEname

The relocated value computer for LO byte relocation is too large or too small to fit into one byte.

E. UNRESOLVED REFERENCE AT nnnn MODULE name FILEname

A reference to an undefined global or section was specified at this point in the object code. This occurs when a global is used in one module but was never defined. The unresolved reference is filled with zeros in the load file.

W. MACHINE REDEFINED FROM microprocessor IN MODULE name FILEname

The current input module has been generated for a different microprocessor than the provious object modules. Differences between microprocessor definitions may cause incompatabilities during linking (e.g., page length, alignment, etc.)

E. SECTION name EXCEEDS MAXIMUM SIZE

Section length is greater than the address space of the microprocessor. The section is not included in the load file. This error may occur when a Reserve is too long.

W. IMPLICIT REORIGIN TO 0 IN SECTION name IN MODULE name FILEname

The Linker processed an object module where code in an absolute Section wrapped around from location FFFFH to 0.

W. SECTION name CHANGED FROM PAGE TO BYTE RELOCATABLE

Either:

1) The section was declared to be page-relocatable and the Linker does not support paging for that microprocessor; or

2) There was insufficient room for a paged section in available memory. The Linker will attempt to allocate memory for the Section on a Byte Relocatable Boundary.

F. (LIST FILE)
(LOAD FILE)
(CONSOLE) I/O ERROR #nn
(COMMAND FILE)
(OBJECT FILE)

This error indicates that the Linker was unable to read to or write from the specified file or device.

W. ATTEMPT TO REFEFINE FILE TYPE FOR filename

Filename was specified twice: Once as an object file and once as a library file. The Linker uses the first file type specified.

LOAD FILE

The primary output from Linker processing is the Load file. A Load file is a subset of the Linker input object modules with all references and relocation resolved.

SYSTEM DIAGNOSTICS

The 9520 Development System Diagnostics are stored on a separate diskette. The diagnostic system provides the user with a comprehensive set of programs and test routines to diagnose and test the 9520 Development System components to isolate malfunctions.

Diagnostic Monitor

The Diagnostic Monitor program is the operating system for the Diagnostic System. It provides the main control for loading and executing test programs. The functions performed by the Diagnostic Monitor are:

- Loads test programs from diskette into memory.
- Provides the keyboard interface routines for selecting and running subtests.
- Provides I/O routines for output to the peripheral port interfaces.
- Provides the option to run tests in sequence or to repeat individual tests.
- Provides general purpose routines used by all of the Diagnostic System programs.

Operation and Running Tests

The description and procedure for loading the system diagnostics and running the various test routines is presented in chapter 9. Refer to this information for using the system diagnostic programs.

TEXT EDITOR, DYNAMIC DEBUGGER AND RELOCATABLE DEBUGGER UTILITY PROGRAMS

TEXT EDITOR UTILITY PROGRAM

The text editor utility for the 9520 Software Development System is WordStar™. In conjunction with the multi-programming monitor control program (MP/M™) operating system, WordStar provides the user with a versatile text editor utility that enhances the 9520 and the user text processor capability for developing, editing, and storing programs. To invoke WordStar, the MP/M operating system must be installed (booted) into the 9520 Development System and the operating system has indicated its readiness with the prompt "0A>" (or 0B> if the user has changed the logged drive to B) displayed on the display terminal. The prompt is given after "booting" at system turn-on, after exiting from WordStar, or after completion of an operating system command.

Invoking WordStar

Once the system prompt "0A>" has been obtained, there are three methods of invoking WordStar:

1. Basic Method: at the prompt "0A>" type:

   ```
   WS@
   ```

   where @ indicates pressing the RETURN key. This starts WordStar with no file being edited; a copyright message appears for several seconds, then the no-file menu (as described in the next subsection) is displayed. Example (underlined text typed by computer):

   ```
   0A>WS@
   ```

   This basic method is sufficient for initial use.

2. To go directly to editing a document: type WS, a space, and the name of the file, including disk drive and type as appropriate. WordStar will proceed to editing this file as though the "edit a document" command had been given from the no-file menu as described below. Examples (underlined text typed by computer, @ indicates carriage return entered by user):

   ```
   0A>WS LETTER.DOC@
   0A>WS B:ABC.XYX@
   ```

3. To go directly to editing a document file with the new file on a different disk drive. This method is for revising extremely long files, where the new file must be placed on a different diskette because of diskette space limitations. Type WS, a space, the name of the file to be edited, another space, and the DRIVE NAME and colon only of the disk drive to receive the edited version of the document. Don't type anything after the destination drive name and colon. Example:

   ```
   0A>WS A:BOOK.DOC B:
   ```

The preceding example says to edit file BOOK.DOC on the diskette in drive A and place the new version on file BOOK.DOC on drive B. When the save is completed, the file on drive A will have been renamed to BOOK.BAK. If a "save and continue edit" command is given, the continuing edit will edit BOOK.DOC from drive B onto drive A; each successive "save and continue edit" will alternate drives.

4. To edit a non-document file, like a file containing program source. Example:

```
WS@
```

Then type N at the no-file menu.

NOTE: If, when you invoke WordStar, you get the following message:

```
You are trying to run an unINSTALLed WordStar.
Please run INSTALL first.
```

then your WordStar has not yet been installed to work with your terminal and printer. Refer to the WordStar User's Guide, Section 14, Installation for installation instructions.

No-File Commands

When started without a file name, or whenever editing of a file is terminated, WordStar displays the "no-file menu": the words "editing no file" are displayed at the top of the screen, and a "menu" of commands that may be entered is shown. Below the menu, WordStar displays the directory (the names of all files on the diskette) of the logged drive if the file directory display is ON. Figure 6-1 shows a typical screen display with the no-file menu:

```
    editing no file

D=create or edit a Document file        H=set Help level
N=create or edit a Non-document file    X=eXit to system
M=Merge-print a file                    P=Print a file
F=File directory off (ON)               Y=delete
L=change Logged disk drive              O=cOpy a file
R=Run a program                         E=rEname a file ¶


DIRECTORY of disk A:
 CHAPTR1.DOC   CHAPTR1.BAK   CHAPTR2.DOC   CHAPTR2.BAK
 CONTENTS      FILE1.DOC     FILE1.BAK     FILE2.DOC
 LETTER.DOC    LETTER.BAK    MERGPRIN.OVR  TEST.DOC
 WS.COM        WSMSGS.OVR    WSOVLY1.OVR
```

Figure 6-1. Screen Showing No-File Menu
(The symbol ¶ represents the cursor position)

To invoke one of the functions shown on the no-file menu, enter the single letter shown for that function. The letter may be entered in upper or lower case, or with the CTRL key depressed. Unrecognized characters are ignored. No RETURN or other key is used after the command letter. When a command is entered, the letter is displayed in the upper left hand corner of the screen and further action is taken depending on the command.

Table 6-1 briefly describes each command; the illustrative examples after the table give further explanation.

Table 6-1. No-File Commands

Command Letter	Function	Description
D	EDIT A DOCUMENT	Asks for file name, then initiates editing of the specified file. The file specified may be an existing file or a new file. To place the new version of the file on a different drive: enter the file name, a space, and the destination drive name followed by a colon.
N	EDIT A NON-DOCUMENT	Same as D except file is edited as a "nondocument", without dynamic pagination and with different defaults.
X	EXIT TO SYSTEM	Exit to MP/M or whatever operating system you are using. Use when you are through with WS and wish to use a system command. The system prompt "A>" will appear next.
H	SET HELP LEVEL	Asks for the new "help level" (0 to 3) which determines the degree of menu display and other prompting supplied by WordStar as will be detailed shortly. Unless the help level is already 0, an explanation of the help levels is displayed.
Y	DELETE A FILE	Asks for file name, then erases file. Performs the same function as the MP/M ERAse console command.
L	CHANGE LOGGED DISK DRIVE	Displays the name of the current logged disk drive and allows selection of a new logged disk drive. Use to allow display of directory of a different drive, or for convenience before working with files on a different drive.
F	FILE DIRECTORY DISPLAY OFF/ON	Controls display of file directory (names of all files on diskette in logged disk drive). First F command turns directory display off, second F turns directory display on again, etc. To display directory of a different drive, change logged disk with L command.
P	PRINT A FILE STOP PRINT CONTINUE PRINT	The P command has three possible effects; depending on whether printing is inactive, a file is being printed, or printing is suspended. The P line in the menu changes as appropriate.

Table 6-1. No-File Commands (continued)

Command Letter	Function	Description
M	MERGE-PRINT A FILE	In order to use the Merge-Print feature of WordStar, the MERGPRIN.OVR file must be present on the disk in drive A. The M command allows merging data from a data file into text at print time for production of form letters, and performs other enhanced print functions. IF MERGPRIN.OVR IS NOT PRESENT, AN ERROR MESSAGE WILL BE DISPLAYED.
R	RUN A PROGRAM	The R command allows you to run a program without exiting from WordStar. For example, the amount of disk space could be checked by using the MP/M program STAT.COM. After R is entered, the following prompt will be displayed: COMMAND? Enter the name of the program to be run and press RETURN. The program name may be followed by file name(s) or other arguments to be used by the program where appropriate.
O	COPY A FILE	The O command allows you to make a copy of a specified file without having to use the MP/M program PIP.COM. You may copy files from or to different disks as long as both disks are on-line at the same time. When O is entered, the following prompts are displayed: NAME OF FILE TO COPY FROM? NAME OF FILE TO COPY TO ? Enter the name of the file to be copied and press RETURN, followed by the name of the file where the copy is to be stored. Specify the disk drive with a drive letter and colon (A:,B:, etc.) preceding the filename.
E	RENAME	The E command allows you to change the name of a file. (E functions like the MP/M command REN.) Enter the name of the file to be renamed followed by the new name in response to the appropriate prompts: Enter NAME OF FILE TO RENAME? NEW NAME?

Illustrative Examples of No-File Commands

D Command: With the no-file menu on the screen, as shown in figure 6-1, type a "D" (or a d or D) to invoke editing of a file. WordStar then displays an explanation and a request to enter the file name as follows:

```
D      editing no file

   Use this command to create a new document file, or to initiate
   alteration of an existing document file.

      A file name is 1 to 8 letters/digits, a period, and an
      optional 1-3 character type.
      File name may be preceded by disk drive letter A-D and
      colon, otherwise current logged disk is used.

   S=delete character     Y=delete entry     F=File directory
   D=restore character    R=Restore entry    U=cancel command

     NAME OF FILE TO EDIT? ¶

   partial DIRECTORY of disk A:   Z=scroll up
    CHAPTR1.DOC   CHAPTR1.BAK   CHAPTR2.DOC   CHAPTR2.BAK
    CONTENTS      FILE1.DOC     FILE1.BAK     FILE2.DOC
```

Figure 6-2. D Command Display
(The symbol ¶ represents the cursor position)

You may then type the desired file name, terminated with a carriage return. The form of a file name is summarized on the screen as a reminder. For example, you might type

```
LETTER.DOC@
```

to edit file LETTER.DOC on the logged drive, or

```
B:LETTER.DOC@
```

to edit the file LETTER.DOC on the diskette in drive B.

To edit file BOOK.DOC on drive A and place the new version of BOOKDOC on drive B, type:

```
A:BOOK.DOC B:@
```

After the carriage return, WordStar proceeds to editing the file, you may then proceed to enter text into the document and/or use WordStar's editing commands. If the file does not exist, NEW FILE is displayed for several seconds. If the NEW FILE appears when you intended to edit an existing file, you probably typed the name wrong or have the wrong diskette in the drive. Abandon the edit (KQ) to get back to the no-file menu.

In figure 6-2 (and also in the screen display of many other WordStar commands) the two lines

```
^S=delete character   ^Y=delete entry   ^F File directory
^D=restore character  ^R=Restore entry  ^U=cancel command
```

remind you of the control characters which may be used to correct typing errors and for other purposes while entering the file name (or other answer). These characters may be used at any time while typing the answer before RETURN is hit. The two display lines appear only at "help levels" 2 and 3, and do not appear until about two seconds have elapsed without a keystroke (this delay is so WordStar will respond faster to fast typists).

To correct a typing error, use control-S (or control-H, DELETE, or BACKSPACE) to delete (erase) characters one at a time. After retyping the desired characters, the control-D key may be used to restore (un-erase) following characters one at a time. For convenience, control-Y erases the entire answer entered, and control-R restores the entire answer or previous answer. For example, if control-R is the first key hit, the name of the last file edited, if any, will appear; it may then be revised (control-S's, retype characters, control-R) if you wish to edit another file with a similar name.

Control-F may be used to invoke the file directory display; Control-F alternately turns the file directory on or off. Control-U may be used to cancel (or abort or interrupt the D command and return to the no-file menu.

NOTE: If the "help level" is zero, the explanatory material shown in figure is omitted from the screen display; only the question "NAME OF FILE TO EDIT" will appear. If you start typing the file name before display of the explanatory material, WordStar will omit some or all of the explanation.

Partial Directory Display: notice that in figure 6-2 the screen shows only part of the disk directory, as indicated by the word "partial". Partial directory display occurs whenever there are more file names than will fit on your screen. To view additional file names, use control-Z to move (scroll) the file directory display up a line, and/or control-W to move the file directory display down a line. Reminders about these control characters -- "^Z=scroll up" and/or "^W=scroll down" -- appear in the line above the directory whenever these characters can be entered and will bring more file names onto the screen.

Y Command. With the No-file menu (figure 6-1) on the screen, type a Y to initiate deletion of a file. The screen display then changes to the following (at help level 3):

```
Y      editing no file

  ^S=delete character    ^Y=delete entry    ^F=File directory
  ^D=restore character   ^R=Restore entry   ^U=cancel command

     NAME OF FILE TO DELETE? ¶

   DIRECTORY of disk A:
    CHAPTR1.DOC   CHAPTR1.BAK   CHAPTR2.DOC   CHAPTR2.BAK
    CONTENTS      FILE1.DOC     FILE1.BAK     FILE2.DOC
    LETTER.DOC    LETTER.BAK    MERGPRIN.OVR  TEST.DOC
    WS.COM        WSMSGS.OVR    WSOVLY1.OVR
```

Figure 6-3. Y Command Display
(The symbol ¶ represents the cursor position)

Enter the name of the file to delete, followed by RETURN. The form of a file name and usage of control characters to correct typing errors, etc. is the same as for the D command (previous example).

After the file is erased, the no-file menu reappears on the screen and another command may be entered. If you enter a Y, then decide not to delete a file, you may cancel the command with ^U, or by hitting RETURN only.

X Command. The X command is used to exit to the operating system MP/M. When an X is typed at the no-file menu, the MP/M prompt (A>) appears at the bottom of the screen.

F Command. The F command turns the file directory display off and on. The first F entered turns the directory display off, the next F restores it, etc. No additional information need be entered, and no screen changes take place except that when directory display is off, the no-file menu item for F changes to read

F=File directory on (OFF)

To display the directory of the diskette in a different disk drive, change the logged drive with the L command (next example). To cause the directory on the screen to be udpated after putting a new diskette in a drive, re-log the same drive with the L command, or press F twice.

L Command. The L command allows changing the logged disk drive. Typing an L at the no-file menu changes the screen display to the following:

```
L       editing no file

  The LOGGED DISK (or Current Disk or Default Disk) is the disk
  drive used for files except those files for which you enter a
  disk drive name as part of the file name.  WordStar displays
  the File Directory of the Logged Disk.

  THE LOGGED DISK DRIVE IS NOW A:

  NEW LOGGED DISK DRIVE (letter, colon, RETURN)?  ¶

   DIRECTORY of disk A:
    CHAPTR1.DOC    CHAPTR1.BAK    CHAPTR2.DOC    CHAPTR2.BAK
    CONTENTS       FILE1.DOC      FILE1.BAK      FILE2.DOC
    LETTER.DOC     LETTER.BAK     MERGPRIN.OVR   TEST.DOC
    WS.COM         WSMSGS.OVR     WSOVLY1.OVR
```

Figure 6-4. L Command Display

To log a different drive, type the letter (A or B; C, D, etc. also acceptable if you have that many drives), a colon, and RETURN. To leave the logged drive unchanged, type control-U, or just hit RETURN.

E Command. The E Command allows you to rename files without having to exit from WORDSTAR. This command performs the same function as the CP/M REN command. Typing an E at the no-file menu will cause the following displayed:

```
E       editing no file

  ^S=delete character    ^Y=delete entry    ^F=File directory
  ^D=restore character   ^R=Restore entry   ^U=cancel command

     NAME OF FILE TO RENAME?  ¶

   DIRECTORY of disk A:
    CHAPTR1.DOC    CHAPTR1.BAK    CHAPTR2.DOC    CHAPTR2.BAK
    CONTENTS       FILE1.DOC      FILE1.BAK      FILE2.DOC
    LETTER.DOC     LETTER.BAK     MERGPRIN.OVR   TEST.DOC
    WS.COM         WSMSGS.OVR     WSOVLY1.OVR
```

Figure 6-5. E Command Display

You may rename a file on another drive by specifying the drive before the name of the file to be renamed (e.g. B:FILENAME.TXT). The NEW NAME? prompt is displayed after the name of the file to be renamed has been entered.

R Command. The R command allows the user to run a different program without exiting from WordStar. This command is especially useful for determining the amount of available disc space by running the CP/M program STAT.COM. When R is entered at the no-file menu, the following prompt is displayed:

```
R      editing no file

  Enter name of program you wish to Run, optionally followed by
  appropriate arguments.
       Example (shows disk space):  STAT

  ^S=delete character     ^Y=delete entry     ^F=File directory
  ^D=restore character    ^R=Restore entry    ^U=cancel command

     COMMAND  ¶

   DIRECTORY of disk A:
    CHAPTR1.DOC   CHAPTR1.BAK   CHAPTR2.DOC   CHAPTR2.BAK
    CONTENTS      FILE1.DOC     FILE1.BAK     FILE2.DOC
    LETTER.DOC    LETTER.BAK    MERGPRIN.OVR  TEST.DOC
    WS.COM        WSMSGS.OVR    WSOVLY1.OVR
```

Figure 6-6. R Command Display

Enter the name of the program to be run (e.g. STAT, to display the amount of space left on a diskette) and press RETURN. Only executable programs (file type .COM) should be specified. An attempt to run a non-executable file may result in an error message, or may or lock up your system making it necessary to re-boot. When the program has completed, the following prompt is displayed:

```
Hit any key to return to WordStar:
```

This allows you to view any results displayed by the program before returning to the WordStar no-file menu.

The R command will handle any CP/M console command (CCP command) command except the resident commands (TYPE, DIR, ERA, REN, and SAVE). File names or other arguments may follow the program name, as in CP/M commands. For example:

```
STAT LETTER.DOC
```

shows the size of file LETTER.DOC on the logged drive. Asterisks and question marks can be used to form "wild card" file names, as in CP/M console commands. For example:

```
STAT B:#.DOC
```

shows the size of all files of type .DOC on the diskette in drive B. (#'s and ?'s are not allowed in file names entered in other WordStar commands).

In order to use the R command, you must have WS.COM (or other name as specified during INSTALLation) on the disk in drive A or the current logged drive.

O Command. The O command provides a way to copy files without exiting from WordStar. When O is entered, WordStar displays the following prompts:

```
O       editing no file

  ^S=delete character     ^Y=delete entry     ^F=File directory
  ^D=restore character    ^R=Restore entry    ^U=cancel command

     NAME OF FILE TO COPY FROM? ¶
     NAME OF FILE TO COPY TO  ?

   DIRECTORY of disk A:
    CHAPTR1.DOC    CHAPTR1.BAK    CHAPTR2.DOC    CHAPTR2.BAK
    CONTENTS       FILE1.DOC      FILE1.BAK      FILE2.DOC
    LETTER.DOC     LETTER.BAK     MERGPRIN.OVR   TEST.DOC
    WS.COM         WSMSGS.OVR     WSOVLY1.OVR
```

Figure 6-7. O Command Display

If the name of an existing file is entered as the file to copy to, WordStar displays the prompt

```
FILE d:name.typ EXISTS -- OVERWRITE? (Y/N):  ¶
```

Press Y (or y or ^Y) to proceed with the copy, destroying the present contents of the copy-to file. Pressing any other key will cause the NAME OF FILE TO COPY TO? question to be reasked; press RETURN or ^U to abort the copy command.

You may copy files from and/or to drives other than the logged drive by specifying a drive before the file name (e.g. B:FILENAME.TXT). The exact filename to be copied must be entered; you may not use asterisks (*) nor question marks. You may use -'s in the file names (with htis or any other WordStar command) if soft-hyphen entry is OFF (OFF is the default) or by entering the - as *P-.

Help Levels

The Help Level setting controls the amount of explanatory material automatically displayed by WordStar, and determines whether and when part of the screen is used while editing to display a "menu" of command keys which may be entered.

The Help Level is initially set to 3, the most helpful level. As you gain experience with WordStar, you will want to reduce the help level in order to have more of the screen available for file display.

The Help Level is changed with the H command on the no-file menu (above), or with the ^JH command while editing a file. Either command displays an explanation of help levels and current help level, and requests a new help level, as shown in figure 6-8.

```
H       editing no file

    HELP LEVELS
     3  all menus and explanations displayed
     2  main editing menu (1-control-char commands) suppressed
     1  prefix menus (2-character commands) also suppressed
     0  command explanations (including this) also suppressed

    CURRENT HELP LEVEL IS 3

    ENTER Space OR NEW HELP LEVEL (0, 1, 2, OR 3):  ¶

    partial DIRECTORY of disk A:  *Z=scroll up
     CHAPTR1.DOC    CHAPTR1.BAK    CHAPTR2.DOC    CHAPTR2.BAK
     CONTENTS       FILE1.DOC      FILE1.BAK      FILE2.DOC
     LETTER.DOC     LETTER.BAK     MERGPRIN.OVR   TEST.DOC
```

Figure 6-8. Help Level Command Display

Unlike the "FILE NAME?" questions asked by the D and Y commands, this question takes a single-key response; no RETURN is needed. Pressing any key other than 0, 1, 2, or 3 leaves the help level unchanged.

NOTE: If you enter the digit (or press any key) before the explanation displays, some or all of the explanation will be ommitted. This provides rapid response for the user that knows what he wants and types, for example, "H2".

The difference between help levels 3, 2, and 1 are manifest primarily when editing a file. Level 0 differs from the higher levels ihn that extra explanations associated with several commands are skipped. The explanations ommitted at help level 0 inlucde, for example, the explanation of the help levels (figure 6-8), the explanations displayed by the D command (6-2), and the explanation of the logged disk drive for the L command (figure 6-4).

The two lines that remind you of the control characters which may be used while answering any question whose prompt ends in a question mark,

```
^S=delete character    ^Y=delete entry    ^File directory
^D=restore character   ^R=Restore entry   ^U=cancel command
```

are displayed above such questions only at help level 2 or 3; the control characters nevertheless work at all help levels.

In addition to the automatically displayed information affected by the help level, WordStar has explicit commands that can be invoked while editing which display information on various subjects. For example, the command *JD (entered while editing) invokes a sequence of screen displays describing the print directives.

Additional information relative to WordStar can be obtained in the WordStar User's Manual.

DYNAMIC DEBUGGER AMD RELOCATABLE DEBUGGER

Dynamic Debugger

The Dynamic Debugger Tool (DDT) program allows dynamic interactive testing and debugging of programs generated in the MP/M environment. The debugger is initiated by typing one of the following commands at the MP/M Console Command level

```
DDT
DDT filename.HEX
DDT filename.COM
```

where "filename" is the name of the program to be loaded and tested. In both cases, the DDT program is brought into main memory in the place of the Console Command Processor (refer to the CP/M Interface Guide for standard memory organization), and thus resides directly below the Basic Disk Operating System portion of MP/M. The BDOS starting address, which is located in the address field of the JMP instruction at location 5H, is altered to reflect the reduced transient program area size.

The second and third forms of the DDT command shown above perform the same actions as the first, except there is a subsequent automatic load of the specified HEX or COM file. The action is identical to the sequence of commands

```
DDT
Ifilename.HEX or Ifilename.COM
R
```

where the I and R commands set up and read the specified program to test (see the explanation of the I and R commands below for exact details).

Upon initiation, DDT prints a sign-on message in the format

```
nnK DDT-s VER m.m
```

where nn is the memory size (which must match the CP/M system being used), s is the hardware system which is assumed, corresponding to the codes

```
D  -  Digital Research standard version
M  -  MDS version
I  -  IMSAI standard version
O  -  Omron systems
S  -  Digital Systems standard version
```

and m.m. is the revision number.

Relocatable Debugger

In addition to the non-relocatable debugging tool (DDT.COM), there is a relocatable debugging tool (RDT.COM). The valid commands for RDT and DDT are the same.

Commands

Following the sign on message, DDT prompts the operator with the character "-" and waits for input commands from the console. The operator can type any of several single character commands, terminated by a carriage return to execute the command. Each line of input can be line-edited using the standard MP/M controls

rubout	remove the last character typed
ctl-U	remove the entire line, ready for re-typing
ctl-C	system reboot

Any command can be up to 32 characters in length (an automatic carriage return is inserted as the 33rd character), where the first character determines the command type

A	enter assembly language mnemonics with operands
B	allows update of bitmap of page relocatable file
D	display memory in hexadecimal and ASCII
F	fill memory with constant data
G	begin execution with optional breakpoints
I	set up a standard input file control block
L	list memory using assembler mnemonics
M	move a memory segment from source to destination
N	relocate a page relocatable file
R	read program for subsequent testing
S	substitute memory values
T	trace program execution
U	untraced program monitoring
V	compute the parameter to follow the W (Write Disk) Command
X	examine and optionally alter the CPU state
W	write a patched program to diskette

The command character, in some cases, is followed by zero, one, two, or three hexadecimal values which are separated by commas or single blank characters. All DDT numeric output is in hexadecimal form. In all cases, the commands are not executed until the carriage return is typed at the end of the command.

At any point in the debug run, the operator can stop execution of DDT using either a ctl-C or G0 (jmp to location 0000H), and save the current memory image using a SAVE command of the form

```
SAVE n filename.COM
```

where n is the number of pages (256-byte blocks) to be saved on disk. The number of blocks can be determined by taking the high-order byte of the top load address and converting this number to decimal. For example, if the highest address in the transient program area is 1234H then the number of pages is 12H, or 18 in decimal. Thus the operator could type a ctl-C during the debug run, returning to the Console Processor level, followed by

```
SAVE 18 X.COM
```

The memory image is saved as X.COM on the diskette, and can be directly executed by simply typing the name X. If further testing is required, the memory image can be recalled by typing

```
DDT X.COM
```

which reloads previously saved program from location 100H through page 18 (12FFH). The machine state is not a part of the OCM file, and thus the program must be restarted from the beginning in order to properly test it.

The individual commands are given below in some detail. In each case, the operator must wait for the prompt character (-) before entering the command. If control is passed to a program under test, and the program has not reached a breakpoint, control can be returned to DDT by executing a RST 7 from the front panel (note that the rubout key should be used instead if the program is executing a T or U command). In the explanation of each command, the command letter is shown, in some cases, with numbers separated by commas, where the numbers are represented by lower case letters. These numbers are always assumed to be in a hexadecimal radix, and from one to four digits in length (longer numbers will be automatically truncated on the right).

Many of the commands operate upon a "CPU state" which corresponds to the program under test. The CPU state holds the registers of the program being debugged, and initially contains zeroes for all registers and flags except for the program counter (P) and stack pointer (S), which default to 100H. The program counter is subsequently set to the starting address given in the last record of a HEX file if a file of this form is loaded (see the I and R commands)

1. A (Assemble) Command - DDT allows inline assembly language to be inserted in to the current memory image using the A command which takes the form

   ```
   As
   ```

   where s is the hexadecimal starting address for the inline assembly. DDT prompts the console with the address of the next instruction to fill, and reads the console, looking for assembly language mnemonics (see the Intel 8080 Assembly Language Reference Card for a list of mnemonics) followed by register references and operands in absolute hexadecimal form. Each successive load address is printed before reading the console. The A command terminates when the first empty line is input from the console.

   Upon completion of assembly language input, the operator can review the memory segment using the DDT disassembler (see the L command).

   Note that the assembler/disassembler portion of DDT can be overlayed by the transient program being tested, in which case the DDT program responds with an error condition when the A and L commands are used.

2. B (Bitmap Bit Set/Reset) Command - The purpose of the BITMAP BIT SET/RESET command is to enable the user to update the bitmap of a page relocatable file. To edit a PRL file the user would read the file in, make changes to the code, and then determine the bytes which needed relocation (e.g., the high order address bytes of jump instructions). The 'B' command would then be used to update the bit map. There are two parameters specified, the address to be modified (0100H is the base of the program segment), followed by a zero or a one. A value of one specifies bit setting.

3. D (Display) Command - The D command allows the operator to view the contents of memory in hexadecimal and ASCII formats. The forms are

```
D
Ds
Ds,f
```

In the first case, memory is displayed from the current display address (initially 100H) and continues for 16 display lines. Each display line takes the form shown below

```
aaaa bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb ccccccccccccccc
```

where aaaa is the display address in hexadecimal, and bb represents data present in memory starting at aaaa. The ASCII characters starting at aaaa are given to the right (represented by the sequence of c's), where non-graphic characters are printed as a period (.) symbol. Note that both upper and lower case alphabetics are displayed, and thus will appear as upper case symbols on a console device that supports only upper case. Each display line gives the values of 16 bytes of data, except that the first line displayed is truncated so that the next line begins at an address which is a multiple of 16.

The second form of the D command shown above is similar to the first, except that the display address is first set to address s. The third form causes the display to continue from address s through address f. In all cases, the display address is set to the first address not displayed in this command, so that a continuing display can be accomplished by issuing successive D commands with no explicit addresses.

Excessively long displays can be aborted by pushing the rubout key.

4. F (Fill) Command - The F command takes the form

```
Fs,f,c
```

where s is the starting address, f is the final address, and c is a hexadecimal byte constant. The effect is as follows: DDT stores the constant c at address s, increments the value of s and tests against f. If s exceeds f then the operation terminates, otherwise the operation is repeated. Thus, the fill command can be used to set a memory block to a specific constant value.

5. G (Go) Command - Program execution is started using the G command, with up to two optional breakpoint addresses. The G command takes one of the forms

```
G
Gs
Gs,b
Gs,b,c
G,b
G,b,c
```

The first form starts execution of the program under test at the current value of the program counter in the current machine state, with no breakpoints set (the only way to regain control in DDT is through a RST 7 execution). The current program counter can be viewed by typing an X or XP command. The second form is similar to the first except that the program counter in the current machine state is set to address s before execution begins. The third form is the same as the second, except that program execution stops when address b is encountered (b must be in the area of the pgoram under test). The instruction at location b is not executed when the breakpoint is encountered. The fourth from is identical to the third, except that two breakpoints are specified, one at b and the other at c. Encountering either breakpoint causes execution to stop, and both breakpoints are subsequently cleared. The last two forms take the program counter from the current machine state, and set one and two breakpoints, respectively.

Execution continues from the starting address in real-time to the next breakpoint. That is, there is no intervention between the starting address and the break address by DDT. Thus, if the program under test does not reach a breakpoint, control cannot return to DDT without executing a RST 7 instruction. Upon encountering a breakpoint, DDT stops execution and types

```
*d
```

where d is the stop address. The machine state can be examined at this point using the X (Examine) command. The operator must specify breakpoints, which differ from the program counter address at the beginning of the G command. Thus, if the current program counter is 1234H, then the commands

```
G,1234
```

and

```
G400,400
```

both produce an immediate breakpoint, without executing any instructions whatsoever.

6. I (Input) Command - The I command allows the operator to insert a file name into the default file control block at 5CH (the file control block created by MP/M for transient programs is placed at this location; see the CP/M Interface Guide). The default FCB can be used by the program under test as if it had been passed by the console. Note that this file name is also used by DDT for reading additional HEX and COM files. The form of the I command is

```
Ifilename
```

or

```
Ifilename.filetype
```

If the second form is used, and the filetype is either HEX or COM, then subsequent R commands can be used to read the pure binary or hex format machine code (see the R command for further details).

7. L (List) Command - The L command is used to list assembly language mnemonics in a particular program region. The forms are

```
L
Ls
Ls,f
```

The first command lists twelve lines of disassembled machine code from the current list address. The second form sets the list address to s, and then lists twelve lines of code. The last from lists disassembled code from s through address f. In all three cases, the list address is set to the next unlisted location in preparation for a subsequent L command. Upon encountering an execution breakpoint, the list address is set to the current value of the program counter (see the G and T commands). Again, long typeouts can be aborted using the rubout key during the list process.

8. M (Move) Command - The M command allows block movement of program or data areas from one location to another in memory. The form is

```
Ms,f,d
```

where s is the start address of the move, f is the final address of the move, and d is the destination address. Data is first moved from s to d, and both addresses are incremented. If s exceeds f then the move operation stops, otherwise the move operation is repeated.

9. N (Normalize) Command - The purpose of the NORMALIZE command is to relocate a page relocatable file which has been read into memory by the debugger. To debug a PRL program the user would read it in with 'R' command and then use the 'N' command to relocate it within the memory segment the debugger is executing.

10. R (Read) Command - The R command is used in conjunction with the I command to read COM and HEX files from the diskette into the transient program area in preparation for the debug run. The forms are

```
R
Rb
```

where b is an optional bias address which is added to each program or data address as it is loaded. The load operation must not overwrite any of the system parameters from 000H through 0FFH (i.e., the first page of memory). If b is omitted, then b=0000 is assumed. The R command requires a previous I command, specifying the name of a HEX or COM file. The load address for each record is obtained from each individual HEX record, while an assumed load address of 100H is taken for COM files. Note that any number of R commands can be issued following the I command to again read the program under test, assuming the tested program does not destroy the default area at 5CH. Further, any file specified with the filetype "COM" is assumed to contain machine code in pure binary form (created with the LOAD or SAVE command), and all others are assumed to contain machine code in Intel hex format (produced, for example, with the ASM command).

Recall that the command

```
DDT filename.filetype
```

which initiates the DDT program is equivalent to the commands

```
DDT
-Ifilename.filetype
-R
```

Whenever the R command is issued, DDT responds with either the error indicator "?" (file cannot be opened, or a checksum error occurred in a HEX file), or with a load message taking the form

```
NEXT PC
nnnn pppp
```

where nnnn is the next address following the loaded program, and pppp is the assumed program counter (100H for COM files, or taken from the last record if a HEX file is specified).

11. S (Set) Command - The S command allows memory locations to be examined and optionally altered. The form of the command is

```
Ss
```

where s is the hexadecimal starting address for examination and alteration of memory. DDT responds with a numeric prompt, giving the memory location, along with the data currently held in the memory location. If the operator types a carriage return, then the data is not altered. If a byte value is typed, then the value is stored at the prompted adress. In either case, DDT continues to prompt with successive addresses and values until either a period (.) is typed by the operator, or an invalid input value is detected.

12. T (Trace) Command - The T command allows selective tracing of program execution for 1 to 65535 program steps. The forms are

```
T
Tn
```

In the first case, the CPU state is displayed, and the next program step is executed. The program terminates immediately, with the termination address displayed as

```
*hhhh
```

where hhhh is the next address to execute. The display address (used in the D command) is set to the value of H and L, and the list address (used in the L command) is set to hhhh. The CPU state at program termination can then be examined using the X command.

The second form of the T command is similar to the first, except that execution is traced for n steps (n is a hexadecimal value) before a program breakpoint occurs. A breakpoint can be forced in the trace mode by typing a rubout character. The CPU state is displayed before each program step is taken in trace mode. The format of the display is the same as described in the X command.

Note that program tracing is discontinued at the interface to MP/M, and resumes after return from MP/M to the program under test. Thus, MP/M functions which access I/O devices, such as the disk drive, run in real-time, avoiding I/O timing problems. Programs running in trace mode execute approximately 500 times slower then real time since DDT gets control after each user instruction is executed. Interrupt processing routines can be traced, but it must be noted that commands which use the breakpoint facility (G, T, and U) accomplish the break using a RST 7 instructin, which means that the tested program connot use this interrupt location. Further, the trace mode always runs the tested program with interrupts enabled, which may cause problems if asynchronous interrupts are received during tracing.

Note also that the operator should use the rubout key to get control back to DDT during trace, rather than executing a RST 7, to ensure that the trace for the current instruction is completed before interruption.

13. U (Untrace) Command - The U command is identical to the T command except that intermediate program steps are not displayed. The untrace mode allows from 1 to 65535 (0FFFFH) steps to be executed in monitored mode, and is used principally to retain control of an executing program while it reaches steady state conditions. All conditions of the T command apply to the U command.

14. V (Value) Command - The purpose of the VALUE command is to facilitate use of the WRITE DISK command by computing the parameter to follow the 'W'. A single parameter immediately follows the 'V' which is the NEXT location following the last byte to be written to disk.

15. X (Examine) Command - The X command allows selective display and alteration of the current CPU state for the program under test. The forms are

```
X
Xr
```

where r is one of the 8080 CPU registers

C	Carry Flag	(0/1)
Z	Zero Flag	(0/1)
M	Minus Flag	(0/1)
E	Even Parity Flag	(0/1)
I	Interdigit Carry	(0/1)
A	Accumulator	(0-FF)
B	BC register pair	(0-FFFF)
D	DE register pair	(0-FFFF)
H	HL register pair	(0-FFFF)
S	Stack Pointer	(0-FFFF)
P	Program Counter	(0-FFFF)

In the first case, the CPU register state is displayed in the format

```
CfZfMfEfIf A=bb B=dddd D=dddd H=dddd S=dddd P=dddd inst
```

where f is a 0 or 1 flag value, bb is a byte value, and dddd is a double byte quantity corresponding to the register pair. The "inst" field contains the disassembled instruction which occurs at the location addressed by the CPU state's program counter.

The second form allows display and optional alteration of register values, where r is one of the registers given above (C, Z, M, E, I, A, B, D, H, S, or P). . In each case, the flag or register value is first displayed at the console. The DDT program then accepts input from the console. If a carriage return is typed, then the flag or register value is not altered. If a value in the proper range is typed, then the flag or register value is altered. Note that BC, DE, and HL are displayed as register pairs. Thus, the operator types the entire register pair when B, C, or the BC pair is altered.

16. W (Write Disk) Command - The purpose of the WRITE DISK command is to provide the capability to write a patched program to disk. A single parameter immediately follows the 'W' which is the number of sectors (128 bytes/sector) to be written. This parameter is entered in hexadecimal.

IMPLEMENTATION NOTES

The organization of DDT allows certain non-essential portions to be overlayed in order to gain a larger transient program area for debugging large programs. The DDT program consists of two parts: the DDT nucleus and the assembler/disassembler module. The DDT nucleus is loaded over the Console Command Processor, and, although loaded with the DDT nucleus, the assembler/disassembler can be overlayed unless used to assemble or disassemble.

In particular, the BDOS address at location 6H (address field of the JMP instruction at location 5H) is modified by DDT to adress the base location of the DDT nucleus which, in turn, contains a JMP instruction to the BDOS. Thus, programs which use this address field to size memory see the logical end of memory at the base of the DDT nucleus rather than the base of the BDOS.

The assembler/disassembler module resides directly below the DDT nucleus in the transient program area. If the A, L, T, or X commands are used during the debugging process then the DDT program again alters the address field at 6H to include this module, further reducing the logical end of memory. If a program loads beyond the beginning of the assembler/disassembler module, the A and L commands are lost (their use produces a "?" in response), and the trace and display (T and X) commands list the "inst" field of the display in hexadecimal, rather than as a decoded instruction.

SYSTEM GENERATION

GENERAL

This chapter describes how to modify the MP/M operating system configuration to meet specific needs for the users application environment.

The need for generating a new system configuration arises when the current space that is allocated for memory segment bases does not accommodate the size of the users program and could result in a possible overlap of memory partitioning for the transient program area. Also, the system operating requirements, such as, the need for additional resident programs, or the add-on of optional extended memory capability would make it necessary to change parameters and generate a new system configuration.

CAUTION: Prior to performing the steps in this chapter, Millennium recommends that the user duplicate the system diskette provided with the 9520. After the 9520 has been power-up (refer to chapter 3), insert the system diskette into the disk drive and press the 9520 front panel RESET switch to reinitialize the 9520 so the software on the diskette is pulled into the system. When the prompt 0A> is displayed on the display terminal, insert the diskette in the unused disk drive, enter the command FDISK <cr> and the diskette duplication procedure will be invoked.

GENSYS PROGRAM DESCRIPTION

The MP/M system generation process employs the GENSYS program which is invoked by the GENSYS command. The program is interactive and displays a table of the current system parameters. The user is prompted to provide inputs which modify the existing system parameters and establish a new system configuration.

The GENSYS program automatically builds a system program file which contains the assigned parameters and writes this program to the system diskette so that the information can be loaded into memory whenever the system is started up from a power-on or reset state.

The system configuration thus established by the GENSYS program remians effective until another GENSYS is performed by the user.

MODIFYING SYSTEM CONFIGURATION

GENSYS is invoked by keying

```
0A <GENSYS> <cr>
```

The system will respond by displaying the following table (Example #1) which reflects the current parameters that were assigned for the system when the last GENSYS was performed. (The example is presented for a Non-Bank Switched System.) The user is prompted to respond to each line item and enter the parameter data shown in brackets.

NOTE: Angle brackets are presented in the example to clarify the user response and are not included in the actual display.

Example #1, System without Bank Switched Memory

```
                    0A> GENSYS

                MP/M System Generation
                ======================

FF              Top page of memory = <ff> or <0>
2               Number of consoles = <2> or <1>
7               Breakpoint RST #   = <6>
N               Add system call user stacks (Y/N)?<y>
Y               Z80 CPU (Y/N)?y
N               Bank switched memory (Y/N)?<n>
                Memory segment bases, (ff terminates list)
                  :<00>
                  :<50>
                  :<a0>
                  :<ff>
                Select Resident System Processes:  (Y/N)
                ABORT    ?<n>
                SPOOL    ?<n>
                MPMSTAT  ?<y>
                SCHED    ?<y>
                MP/M
                0A>
```

The description of each line item response entered during system generation is as follows:

(1) Top Page of Memory

Two hexadecimal ASCII digits are to be entered giving the top page of memory. A value of 0 can be entered in which case the MP/M loader will determine the size of memory at load time by finding the top page of RAM.

(2) Number of Consoles

Each console specified will require 256-bytes of memory. The 9520 Development System supports up to two consoles.

(3) Breakpoint RST #

The breakpoint restart number to be used by the SID (Symbolic Instruction Debugger) and DDT (Dynamic Debugging Tool) debuggers is specified. Restart 0 is not allowed.

(4) Add System Call User Stacks (Y/N)?

If you desire to execute CP/M *.COM files then your response should be Y. The Y response forces a stack switch with each system call from a user program. MP/M requires more stack space than CP/M.

(5) Bank Switched Memory (Y/N)?

If your system does not have bank switched memory than you should respond with a N. Otherwise respond with a Y along with the additional questions and responses (as shown in the Example #2 table) which will be required.

(6) Memory Segment Bases

Memory segmentation is defined by the entries which are made. Care must be taken in the entry of memory bases as all entries must be made with successively higher bases.

(7) Select Resident System Processes (RSPs)

A directory search is made for all files of type RSP. Each file found is listed and included in the generated system file if you respond with a Y.

where: ABORT: Allows the user to abort a running program

SPOOL: Allows the user to spool ASCII text files to the list device

MPMSTAT: Allows the user to display the run-time status of the operating system

SCHED: Allows the user to schedule a program for execution

After the last user response is keyed, the GENSYS program produces the MPM.SYS file which contains the new parameters assigned for the system configuration.

The MPM.SYS file is then written to the system diskette and is booted into memory whenever the system is started up from a power-on or reset state.

The updated MP/M System Generation Table is displayed on the screen and the prompt (0A>) is presented so that operator commands can be issued.

BANK SWITCHED MEMORY CONFIGURATION

Bank switched memory is the condition where the add-on, 48K expansion memory bank is available to operate in parallel with the base memory bank. The user can specify the segmenting of transient spaces in both memory banks.

When the system is configured with the Bank Switched Memory and Banked BDOS (Basic Operating System) File manager, the user is prompted to provide additional inputs to satisfy these configuration requirements.

This procedure requires an initial GENSYS and MPMLDR execution to determine the exact size of the operating system, followed by a second GENSYS.

The system will display the following table (Example #2) so that the user can specify the bank switched memory requirements. The table in Example #2 is abbreviated to emphasize system prompts for the bank switched memory. (Note: Angle brackets are presented in the example to clarify the user responses and are not included in the actual display.)

Example #2, System Generation with Bank Switched Memory

```
0A> GENSYS

 MP/M System Generation
 ======================

Top page of memory = <ff>
.                       .
.                       .
.                       .
.                       .
Bank switched memory (Y/N)?<y>
Banked BDOS file manager (Y/N)?<y>
Enter memory segment table:  (ff terminates list)
 Base,size,attrib,bank = <0,50,0,0>
 Base,size,attrib,bank = <ff>
.                 .
.                 .
.                 .
SCHED     ? <y>
MP/M
0A>
```

The description of each line item response entered for Bank Switched Memory during system generation is as follows:

(1) Bank Switched Memory

Respond with a Y if the system hardware is configured with the add-on, 48K expansion memory.

(2) Bank Switched BDOS File Manager

Always respond with a Y to include the bank switched BDOS. This will provide an additional 0C00H bytes of common area for some RSPs (resident system processes). The banked BDOS is slower than the non-banked because FCBs (file control blocks) must be copied from the bank of the calling program to common and then back again each time a BDOS disk function is invoked.

Memory Segment Table

When bank switched memory has been specified, you are prompted for the base, size, attributes, and bank for each memory segment. Extreme care must be taken when making these entries as there is no error checking done by GENSYS regarding this function.

The first entry made will determine the bank in which the banked BDOS is to reside. It is further assumed that the bank specified in the first entry is the bank which is switched in at the time the MPMLDR is executed.

The attribute byte is normally defined as 00.

The bank byte value is hardware dependent and is 0 for bank 0 and 1 for bank 1.

Next, execute the MPMLDR in order to obtain the base address of the operating system. The base address in this example will be the address of BNKBDOS.SPR (BC00H) as shown in Example #3 table.

Example #3, Base Address Assignment for Bank Switched Memory

```
0A>MPMLDR

MP/M Loader
===========

Number of consoles  =  2
Breakpoint RST #    =  6
Z80 CPU
Banked BDOS file manager
Top of memory       = FFFFH

Memory Segment Table:
SYSTEM    DAT   FF00H   0100H
CONSOLE   DAT   FD00H   0200H
USERSYS   STK   FC00H   0100H
XIOS      SPR   F600H   0600H
BDOS      SPR   EE00H   0800H
XDOS      SPR   CF00H   1F00H
Sched     RSP   CA00H   0500H
BNKBDOS   SPR   BA00H   0E00H
--------------------------
Memseg    Usr   0000H   5000H   Bank 00H
```

Using the information obtained from the initial GENSYS and MPMLDR.execution the following GENSYS can be executed to reflect the bank switched memory configuration:

```
   0A>  <GENSYS> <cr>

MP/M System Generation
======================

Top page of memory  = <ff>
Number of consoles  = <2>
Breakpoint RST #    = <6>
Add system call user stacks (Y/N)?<y>
Z80 CPU (Y/N)?<y>
Bank switched memory (Y/N)?<y>
Banked BDOS file manager (Y/N)?<y>
Enter memory segment table: (ff terminates list)
 Base,size,attrib,bank = <0,ba,0,0>
 Base,size,attrib,bank = <0,c0,0,1>
 Base,size,attrib,bank = <ff>
Select Resident System Processes:  (Y/N)
ABORT   ?<n>
SPOOL   ?<n>
MPMSTAT ?<n>
SCHED   ?<y>
MP/M
0A>
```

SYSTEM OPERATION

INTRODUCTION

Operation of the 9520 Development System generally involves the following types of user activity:

1. Applying AC power at the equipment to begin system operations.
2. Inserting the system diskette and loading the MP/M operating system.
3. Entering commands at the Keyboard to load and execute desired programs from the system diskette and user diskette as appropriate.
4. Creating and maintaining source files for user programs.
5. Aborting and detaching from a running program.
6. Closing out files and removing diskettes to shut down system operations.

User input and interaction is implemented through the terminal console keyboard and display screen. The commands and data entered at the keyboard are concurrently displayed on the screen along with system responses.

The system generates output to the various system devices, floppy disk drives, display terminal and I/O interface channels at the RS-232, RS-422, and IEEE-488 ports.

TERMINAL KEYBOARD AND KEY FUNCTIONS

The terminal keyboard is generally arranged like that of a typewriter, except for a number of additional keys. The placement and labels on keys will vary for different keyboards. A typical keyboard arrangement showing all essential keys for using the 9520 Development System is shown in figure 8-1. These keys can be grouped into three functional categories as follows:

- o Data Entry Keys
- o Program Control Character Keys
- o Text Control Character Keys

Data Entry Keys

Data Entry keys are used to provide user input to, and interaction with the system. After keying in the desired input, the RETURN key must be pressed -- only then does the system accept the input. There are some exceptions, however, such as control character inputs to certain system responses which do not require the RETURN input.

The EDIT utility program accepts both upper and lowercase ASCII characters as input from the keyboard. Single letter commands such as control characters may also be typed as either lower or upper case.

The data entry keys are similar to a standard typewriter keyboard. These keys provide 96 ASCII characters including upper and lower case English alphanumeric and special characters. It also includes the control keys for DEL, CTRL, ESC and RETURN functions.

The DEL key is used to delete the last character which was typed at the keyboard. The DEL key may also be labeled DELETE, RUB or RUBOUT and may, or may not, be shared with the underscore key and may, or may not, require SHIFT to activate. The BACKSPACE key or the key's control-H may be used for the same purpose.

The escape key will have no effect on MP/M or CP/M.

The RETURN key may be labeled CARRIAGE RETURN or ENTER and causes the cursor to be moved to column 1 of the current line.

Note that the space bar is for entering spaces. Unlike on a typewriter, the space bar cannot be used to move across characters already displayed on the screen.

Figure 8-1. Typical Keyboard Arrangement

The CTRL key is used like a shift key to enter alphabetic control characters. To type a control character, hold the CTRL key down while typing the letter. In this manual, a caret (^) character is used in front of the letter to indicate a control character. For example, ^D indicates control-D, which is typed by holding down the CTRL key and typing a D; ^B means Control-B, etc. Control characters are used to enter commands for program control and text control during data entry. It is not necessary to use the RETURN key after entering a control character.

A number of additional keys may be present for use in data entry which include the following:

BACKSPACE: Same as control character, ^H which is used for backspacing the cursor on a line.

TAB: Same as control character, ^I which is used with the Text Editor Utility for tabbing.

LINEFEED: Same as control character, ^J which is used to terminate the current input and cause the cursor to move down one line on the screen.

REPEAT: Used to automatically send the same character continuously. Some keyboards will repeat any character whose key is held down 1/2 second. Other keyboards require the REPEAT key to be held down while another key is pressed to send the character string.

CURSOR DIRECTION: Consists of four keys with arrows to indicate the four directions of cursor motion. Pressing the key causes the cursor to move in the direction indicated by the arrow.

Cursor motion can also be initiated by control characters used in the WordStar Text Editor Utility as follows:

^D: Moves cursor to the right -- to the next character in a line.

^S: Moves cursor to the left -- to the previous character in a line.

^E: Moves cursor up to preceeding line on screen.

^X: Moves cursor down to next line on screen.

Program Control Character Keys

The program control characters provide limited user control over the program operation processes, such as:

- Aborting a Program
- Detaching from a Running Program

- Terminating the Current Input
- Deleting and Inserting a Line
- Obtaining Exclusive use of a Listing (printer) Device
- Stopping the Display Output Before Continuing with Execution

The program control characters also permit limited editing functions of a line entry while typing in a command line at the keyboard:

The following control characters are used in the MP/M or CP/M system:

MP/M, CP/M CONTROL CHARACTER	DESCRIPTION OF CONTROL FUNCTION
^C	Abort program in process and terminate execution
^D	Detach from a running program (MP/M only)
^E	Physical end of line
^H	Delete the last character typed at keyboard and backspace one character position
^J	Terminate current input (same as line feed)
^M	Terminate input (same as carriage return)
^R	Retype current command line to provide a clean line following character deletion with rubouts
^U	Remove current line after new line
^X	Delete entire line typed at keyboard and backspace to the beginning of current line
^Z	End input from keyboard

(Control characters ^P, ^Q and ^S affect the console output as follows:)

^P	Copy all remaining keyboard outputs to the list (printer) device. Output is sent to both the list device and display device until the next ^P is typed. If the list device is not available, a PRINTER BUSY status message is displayed on the screen.
^Q	Obtain ownership of the PRINTER BUSY status message (MP/M only). Attaching the printer via this command prevents other terminal consoles from gaining access to the printer by issuing ^P, ^Q, PIP, or SPOOLER commands. The printer is thus owned by the console which issued the ^Q command until another ^P or ^Q command is entered to release the printer.

Control Characters in the MP/M, CP/M System, continued

MP/M, CP/M CONTROL CHARACTER	DESCRIPTION OF CONTROL FUNCTION
	The ^P command should be used only when a program (such as CP/M *.COM file) is executed which does not cause the PRINTER BUSY status message to occur prior to accessing the printer. If the printer is not available, a PRINTER BUSY message is displayed on the screen (MP/M only).
^S	Temporarily stops the display motion. The program execution and display motion will resume after any character is typed on the keyboard. This control is used to halt the display motion on high-speed displays so that the operator can view a segment of the output data before continuing the processing.

Text Control Character Keys

The WordStar™ Text Editor Utility provides an array of text control characters that are fully described in chapter 6.

These text control characters are used in conjunction with the CTRL key to issue a specific command that controls the processing for text inputs, such as, creating a new document file, editing an existing document file, manipulating the display, and listing of document files. Text control functions involve the following categories of commands that can be executed by the text control characters:

- Cursor Motion -- Forward, Backward, Upward and Downward directions
- Scrolling -- Single line, full screen, continuous, up/down and fast/slow motion
- Text Entry -- Insertion, tabbing and tabulating
- Text Deletion on a Page -- Individual characters, words, lines and blocks of data
- Saving and Deleting -- Updated files, original files and old files
- On-Screen Text Formatting -- Set margins, line spacing, tabs and paragraphs
- Find and Replace -- Global File, Word, Phrase, Character Strings
- Place Markers -- Set mark locations in a file for future reference
- Block Commands -- Moving paragraphs, sentences, copying and deleting blocks
- Document or non-document files

- Help Commands -- Display learning aids and reference user information on screen
- Miscellaneous Commands -- Repeat a command, interrupt a command/printing and display file directory

USER INTERACTION

This section describes user interaction with the 9520 Development System to initiate the system from a cold start, load the operating system from the diskette and enter commands at the terminal keyboard. This information is divided into the following subsections:

- System Initiation
- Command Interpreter State
- Immediate Command Mode
- Interactive Command Mode

System Initiation

Use the following step-by-step procedures to initialize the system from a cold start:

1. Turn on the POWER switch at the display terminal.

   a. On some terminals a bell will beep within 1 second to indicate power is on.

   b. After 10 to 15 seconds, the cursor will appear in the upper left-hand corner of the screen and the status line will appear across the bottom of the screen. At this time, the operator can adjust the contrast to obtain the desired brilliance for the screen.

2. Verify the diskette is not mounted on the disk drives and turn on the POWER switch at the 9520 Development System control panel.

   a. After a few seconds, the terminal screen will display the following message to indicate the 9520 Boot PROM, Self Test diagnostic is running:

      9520 SELF TEST VERS 2.0 C0

   b. After the diagnostic test is completed, the following message is displayed on the terminal screen to inform the operator that system software can be loaded:

      9520 SELF TEST VERS 2.0 COMPLETE

3. Insert the system diskette in either of the disk drives (Drive A or Drive B).

   a. Press the latch bar to open the drive access door at the selected drive.

   b. Position the diskette as shown in figure 3-1 and push the diskette forward until a click is heard.

   c. Close the drive access door to lock the diskette on the drive spindle.

NOTE: The subsequent reset procedure need not be performed when the 9520 SELF TEST VERS 2.0 COMPLETE message is displayed.

4. Press the RESET switch on the 9520 Development System control panel to start up system operations and load the operating system from the diskette.

   a. The system will perform the self-test diagnostic and then load the bootstrap loader program from the diskette, which in turn loads the operating system from the diskette.

   b. After a few seconds, loading is completed and the following message appears on the display screen:

```
MP/M
XA >
```

   where: XA shown in prompt:

   X = User code and console assignment number. For example, console #0 is initialized to user #0 for a single user/console system configuration and would be presented as:

```
0A >
```

   Variations for assigning multiple users and consoles to the system are described in section 1.3 of the MP/M Users Manual.

   A = Default for the disk drive (A or B) that is currently logged to the console and can be changed at any console as described in section 1.3 of the MP/M Users Manual.

   The specified default disk must contain the utility files, such as, DIR, REN, ERA, etc.

5. After the inital loading of the MP/M operating system is completed, perform the System Generation Procedure described in chapter 7 to select the operating system parameters. This will place the system in the Command Interpreter State, so that the system is ready to accept and interpret commands that are issued from a local console keyboard or process UPLOAD/DOWNLOAD commands that are issued from a remote emulator/debug station. All keyboard commands are entered as a single-line entry in one of two modes which follow.

Entering Commands

the command may be keyed as an explicit entry (for the Immediate Command Mode) by specifying all of the parameters in the command prior to execution; or the command may be keyed as an implicit entry (for the Interactive Command Mode) so that the user can interact with, and respond to queries issued by the system to provide parameter inputs during the execution. Both types of command entries are described in subsequent paragraphs along with examples.

Document Conventions

The following documentation conventions are used in this manual for describing the command entry syntax -

o Angle Bracket: < >	Item inside angle bracket is a required entry
o Bracket: [ ]	Item inside bracket is optional entry
o Logic OR Symbol: |	Items on either side of symbol may be used in the entry
o Three consecutive Dots: ...	Indicates multiple occurrences of a preceding item

Command Interpreter State is the operating state that allows the system to accept user commands for interpretation and subsequent action.

The Command Line Interpreter is invoked whenever:

o The system is initiated or reset

o The user/system program completes processing a command or terminates execution due to an error

In most cases, the Command Line Interpreter State is indicated by the presence of the prompt character, XA>, followed by the cursor in column 2 of the current line on the display screen. The X equals any number from 1 through 15.

Once the system has prompted the user, commands can be entered in one of the following modes -

1. Immediate Command Mode

2. Interactive Command Mode

Immediate Command Mode

The Immediate Command Mode allows entry of one-line commands. The parameters are included in the line entry to identify operators and operands that are required to complete the program execution without prompting the user for additional input.

Syntax for an Immediate command is as follows:

<command> <parameter-1>...<parameter-n> <cr>

where:

<command>	-- identifies a 9520 system utility name, an MP/M or CP/M system utility name, or a user program name (e.g., utility name could be BRATE)
<parameter-1> thru <parameter-n>	-- defines parameters required to run the program. Each parameter may consist of a file name, a function, a device name, an indicator or an assigned value. (e.g., function parameter #1 could be I/O device: PRINTER, parameter #2 could be arithmetic indicator: = parameter #3 could be value for Baud Rate)
<cr>	-- indicates a carriage return is required to complete the command entry

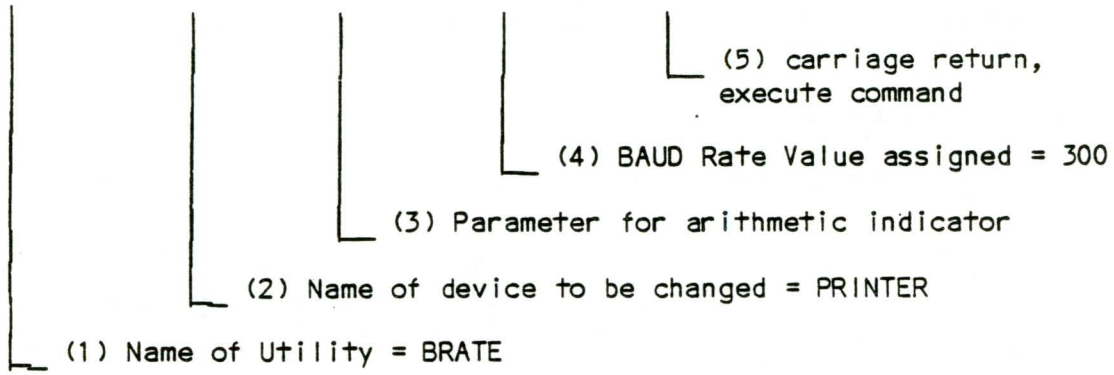
Entering Immediate Command

The Immediate Command can be keyed by using the following techniques for making the entry:

- Use data entry keys to key in characters (all data entry keys are operative).
- Corrections may be made at any time before pressing the RETURN key.
- Make corrections, if necessary, by erasing (using the DELETE, RUB, RUBOUT, BACKSPACE or control H character key) each of the characters up to, and including the first erroneous character in the command, and then re-entering them correctly. The control X character may be used to erase the entire line and start a new line entry.
- Enter each parameter to define operators and operands for the program.
- Press the RETURN key. This initiates execution of the selected program command.

EXAMPLE: Use of single-line Immediate Command entry to call out the Baud Rate Utility and assign a new Baud Rate value for I/O device.

Syntax: <BRATE> <I/O Device> <=> <BAUD Rate> <cr>



Keyboard Entry: BRATE PRINTER = 300 <cr>

Interactive Command Mode

The Interactive Command Mode allows the user to call out a utility and respond to prompts issued by the system after the program begins execution. Each of the prompts will query the user to provide additional input information so that execution can be resumed and/or completed.

The following types of input information is requested by the system prompts, each of which is followed by a carriage return:

- o Parameter #1 (device name, file name, type of function)
- o Parameter #2 (Indicator, Operator)
- o Parameter #3 (Value, address location operand)

Syntax for the Interactive Command sequenced by steps is as follows:

Step 1.	Call out program:	<command> <cr>
Step 2.	Respond to prompt, enter 1st parameter:	<parameter #1> <cr>
Step 3.	Respond to prompt, enter 2nd parameter:	<parameter #2> <cr>
	. . . .	
Step 4.	Respond to prompt, enter last parameter:	<parameter #n> <cr>

Where each entry step performs the following functions:

(1) <command> <cr> -- identifies a 9520 System Utility name, an MP/M or CP/M System Utility, or a user program name that begins executing when carriage return is entered.

(2) <parameter #1> <cr> -- prompts the user to input specific information to satisfy program processing requirements and resume execution.

(3) <parameter #2> <cr> -- prompts the user to input additional information to satisfy program processing requirements and resume execution.

(4) <parameter #n> <cr> -- prompts the user to input final information so that processing will be completed.

Entering Interactive Command

The Interactive Command can be keyed by using the following techniques for making the entry:

- Use data entry keys to key in characters (all data entry keys are operative.)
- Corrections may be made at any time before pressing the RETURN key.
- Make corrections in the same manner described for Immediate Command Entry. The C character may be used to abort the program at any time.
- Enter each parameter as requested by the various prompts to define specific operators and operands for the program. Always press the RETURN key after each parameter entry to resume or complete the execution.

EXAMPLE: Use the multiple-line, Interactive Command Entry to call out the Baud Rate Utility and assign a new Baud Rate value for I/O device:

SYSTEM PROMPT	SYNTAX: USER ENTRY
(1) 0A>	<BRATE> <cr>
	└ carriage return, begin execution
	└ Name of Utility, BRATE
(2) I/O DEVICE:>	<PRINTER> <cr>
	└ carriage return, resume execution
	└ I/O device selected
(3) BAUD RATE:>	<300> <cr>
	└ carriage return, complete execution
	└ Baud Rate Value assigned

Keyboard Entry

```
BRATE  <cr>
>  PRINTER  <cr>
>  300 <cr>
```

RUNNING A PROGRAM

Starting the execution of a program is initated by keying the Program Name followed by a carriage return <cr>. Some programs contain one or more parameters which follow the program name on the line entry. The programs that are provided with MP/M are described in sections 1.4 and 1.5 of the MP/M Users Manual.

ABORTING AN ATTACHED PROGRAM

A program can be aborted by keying a control C (C) character at the console. The C terminates execution of the program which was initated (and thereby attached) by the console. A detached program (i.e., initated from another console) cannot be aborted with the C. A detached program must first be attached and then aborted. A running program may also be aborted using the ABORT command as described in section 1.5 of the MP/M Users Manual.

DETACHING FROM A RUNNING PROGRAM

Detaching from a running program may be invoked by keying a control D (D) character at the console.

In order to detach a program using the D character, the executing program must be performing a check console status to observe the detach request.

ATTACHING TO A DETACHED PROGRAM

A detached program (i.e., the program is not owned by a console) may be attached to a console by keying: ATTACH, followed by the program name. A program may only be re-attached to the console form which it was detached.

SYSTEM DIAGNOSTIC PROGRAM

GENERAL

In addition to the boot PROM, self-test feature permanently installed in the 9520 Software Development System (refer to chapter 3), a diagnostic program is provided on a separate diskette. This paragraph describes the functions of the system diagnostic program.

The Diagnostic Diskette contains software that will thoroughly test all functions of the system. In the event a subassembly fails the diagnostic test, the diagnostic program will cause an error message to be displayed. The error message will identify the cause of failure.

DIAGNOSTIC EXECUTION

Insert the Diagnostic Diskette in disk drive A or B. The contents of Diagnostic Diskette are loaded into the 9520 System by entering the command, DIAG on the console. After the diagnostic is loaded and initialized, it will log on to the display terminal and display:

```
9520 DIAGNOSTIC VERSION 1.0
```

and then the prompt

```
0A>
```

The user can now communicate with the monitor and direct the flow of events by using the commands provided to invoke the various test routines. All commands have the generalized format:

```
xx<=
xxdd...d<=
xxdd...d:xxdddd:xx<=
```

where:

xx = Command mnemonic
dd = Individual data field commands
< = Carriage return

Format - The format xx<= is used when the user requires additional information to implement the command (i.e., the command SD<= will display all tests available for execution) or no data field is necessary (i.e., LT<= will invoke the loop-on-test option).

General Operator Control - The user may temporarily suspend message output (or test execution whenever a message is encountered) by pressing the console SPACEBAR. Testing and message display can be resumed by pressing the SPACE BAR again. The user may terminate test execution by entering x on the console keyboard. This comand causes the diagnostic to return control to the diagnostic monitor program. The prompt 0A> is displayed on the display terminal.

NOTE: There may be a delay before the monitor prompt appears after entering the x command.

COMMANDS

HP (Help) Command - The help command will cause a menu of valid diagnostic monitor commands to be displayed. The help command format is:

```
HP <=
```

Table 9-1 lists the response to the HP command, as displayed on the display terminal.

Table 9-1. Valid Diagnostic Commands

MNEMONIC	DEFINITION
HP	Help-Display Commands
WS	Warm Start
SD	Select Diagnostic
DR	Drive Select
LT	Set Loop-on-Test (S) Option
SM	Set No Messages Option
LE	Loop on Error
SO	Set Operator Intervention Tests Option
HE	Set Halt on Error Option
TL	Set Long Test Option
DP	Display Pass and Error Counts
CO	Continue Testing

```
0A> (Prompt)
```

WS Command - The warm start command will re-initialize the diagnostic to the default conditions with all options reset, and Disc Drives A and B selected for testing. All PASS and ERROR COUNT fields are cleared to zero.

The command format is:

```
WS <=
```

Terminal display response is

```
9520 DIAGNOSTIC VERSION 1.0

0A>
```

SD (SELECT DIAGNOSTIC Test) Command - This command allows the operator to either queue a test (or tests) for execution or have a display of all possible tests.

The command format is:

```
SD <=
```

The display terminal responds with

```
ID                  TEST

01--------RAM TEST-FIXED PATTERNS
02--------RAM TEST-ADDRESS PATTERN
03--------SERIAL I/O PORT 1 TEST
04--------SERIAL I/O PORT 2 TEST
05--------SERIAL I/O PORT 3 TEST
06--------60 Hz TIMER TEST
07--------DISC RESTORE TEST
08--------DISC SEEK TEST
09--------DISC FIXED PATTERNS TEST
10--------DISC TRACK AND SECTOR DATA TEST
11--------DISC RANDOM TRACK-SECTOR-DATE TEST
12--------DISC INTERRUPT NOT READY-READY TEST
13--------DISC INTERRUPT READY-NOT READY TEST
14--------DISC WRITE PROTECT TEST
0A>
```

NOTE: The SO command option must be enabled to perform test 12, 13, and 14.

When the SD command is followed by an asterisk (*), the diagnostic monitor selects and executes all available tests.

When the SD command is followed by nn, the user selects the test (nn) for execution.

The command SD nn1 nn2 nn3 nn5 nn4...nnn <= depicts that any number and/or sequence of tests may be queued for execution.

NOTE: If the SD command is part of a multiple command line string, it must be the last command in the line.

DR (DRIVE SELECT) Command - This command allows the operator to select which disc drive(s) will be used for testing.

The command formats are:

```
DR A <=
```

Selects Drive A

```
"DRIVE A SELECTED"
 0A>
```

is displayed.

```
DR B <=
```

Selects Drive B

```
"DRIVE B SELECTED"
 0A>
```

is displayed.

```
DR * <=
```

Selects both Drive A and B

```
"DRIVE A AND B SELECTED"
 0A>
```

is displayed.

LT (LOOP on TEST(s) Command - The LT command sets the loop-on-test(s) options and causes the test(s) to be executed continuously until the operator suspends execution or an error is encountered if the Halt-on-Error option is set.

The command format is:

```
LT <=
```

The display terminal responds with the prompt

```
0A>
```

SM (SUSPEND MESSAGES) Command - The SM command suspends all messages except the header message of the test(s) selected. This command is intended to be used in conjunction with the loop-on-test or loop-on-error options for troubleshooting purposes.

The command format is:

```
SM <=
```

The display terminal responds with the prompt

```
0A>
```

SO (SET OPERATOR intervention test(s)) Command - This command allows the execution of tests requiring operator intervention within the test(s) selected. The SO command must be set for tests 12, 13, and 14.

The command format is:

SO <=

The display terminal responds with the prompt

0A>

LE (LOOP-ON-ERROR) Command - The LE command will cause the test selected to loop on an error condition (when encountered) without changing any parameters that caused the error. This option is valid only in tests 01, 02, 03, 04, and 05.

The command format is:

LE <=

The display terminal responds with the prompt

0A>

HE (HALT on ERROR option) Command - The HE command causes the selected test execution to halt when an error is encountered. The test execution will continue when the operator types " <= " on console.

The command format is:

HE <=

The display terminal responds with the prompt

0A>

TL (LONG TEST option) Command - When the TL command is set, the DISC FIXED PATTER tests are executed in long form by testing all tracks.

The command format is:

TL <=

The display terminal responds with the prompt

0A>

DP (DISPLAY PASS and ERROR COUNT) Command - This command allows the operator to have the PASS and ERROR COUNT tally displayed on the terminal.

The command format is:

DP <=

The display terminal responds with

```
<TEST> <PASS> <T E> <DRV A> <DRV B> <S E> <RNF E> <CRC E> <DATA E>

01      0000   00
02      0000   00
03      0000   00
04      0000   00
05      0000   00
06      0000   00
07      0000   00    00      00      00      00      00      00
08      0000   00    00      00      00      00      00      00
09      0000   00    00      00      00      00      00      00
10      0000   00    00      00      00      00      00      00
11      0000   00    00      00      00      00      00      00

0A>
```

The label definitions are

<T E>	= Total Errors
<DRV A>	= Total Errors, Drive A only
<DRV B>	= Total Errors, Drive B only
<S E>	= Seek Errors
<RNF E>	= Record-not-found Error
<CRC E>	= CRC Error
<DATA E>	= Data Mismatch Error

All counters are in Hex.
All error counters terminate count at OFF (hex).
All pass counters terminate count at OFFF (hex).

CO (CONTINUE Execution of Suspended Test) Command - The CO command will cause the monitor to resume execution of a selected test that was terminated (suspended) by the user implementing the "X" (EXIT) funciton.

The command format is:

```
CO <=
```

The display terminal responds with the prompt

```
0A>
```

FUNCTIONAL TEST DESCRIPTION

Test ID 01 RAM TEST - Fixed Pattern

A write/read/verify operation os performed on all available memory using the following fixed patterns and a "Walking 1's" pattern

00
FF (Hex)
55 (Hex)
AA (Hex)

If the system contains the optional expansion memory option, it will be tested also.

Test ID 02 RAM TEST - Address Pattern

All available memory is written with data which is equal to the address HI BYTE LOW BYTE. After a period of delay, to ensure refresh is functional, the memory is read and verified. Expansion memory option (if installed) is tested also.

Test ID 03 SERIAL I/O Port 1 Test

Serial Port 1 is tested by enabling the wrap-back function on the 9520 control board and writing the fixed patterns as described in Test 01 (RAM TEST). The Serial port is then read and verified. In addition to the data write/read/verify, the baud rate is tested by measuring the time between received character available.

This test is performed using the following baud rates.

75, 110, 134.5, 150, 300, 600, 1, 2k, 2.4k, 4.8k, 9.6k, 19.2k, 38.4k, 56k, 76.8k

Test ID 04 SERIAL I/O Port 2 Test

Essentially the same as Port 1 with exception of different baud rates of:

75, 110, 134.5, 150, 300, 600, 1.2k, 2.4k, 4.8k, 9.6k

Test ID 05 SERIAL I/O Port 3 Test

Essentially the same as Port 2 except that Port 3 is tested.

Test ID 06 60 Hz TIMER Test

A software timing measurement is performed on the 60 Hz timer interrupt.

Disc Drive/Controller Tests

NOTE: All Disc Tests are performed on Drive A or Drive B or both as specified by operator input. Refer to the "DR" command.

Test ID 07 DISC RESTORE Test

The floppy controller is issued a RESTORE/VERIFY command and the controller status and track register is checked to ensure that track 0 was reached and the track record was read correctly.

Test ID 08 DISC SEEK Test

Using the STEP-IN and STEP-OUT commands, the SEEK operation is performed from TRACK 0 to TRACK 76, sequentially, with a verify performed at each Track. The test then performs SEEKs in the following sequences.

TRACK 76 to 1
TRACK 1 to 75
TRACK 75 to 2, etc.
TRACK 38 to 39

Test ID 09 DISC FIXED PATTERNS Write/Read

All tracks and sectors are written and then verified with the following fixed patterns: 00; FF; 55; AA.

In the long test, the tracks are written sequentially from 0 to 76.

Test ID 10 DISC WRITE-READ TRACK/SECTOR Data Test

All tracks/sectors are written with a 2-byte "word" equal to the track and sector "address". All tracks are written sequentially from Track 0 to Track 76. After all tracks/sectors are written, the disc is read and verified.

Test ID 11 DISC RANDOM TRACK/SECTOR Data Test

This test performs a Write/Read/Verify operation to a random Track/Sector selection with Random data. This sequence is performed 256 times per test pass.

Test ID 12 DISC INTERRUPT on NOT READ/READY Test

This is an operator intervention test which verifies that an interrupt is generated when a drive goes from a NOT READY to READY state. This should occur whenever the operator places a diskette in the drive-under-test and closes the drive door.

Test ID 13 DISC INTERRUPT on READY/NOT READY Test

Essentially the same as above except that the interrupt tested is that which is generated when the operator opens the drive-under-test door and causes the NOT READY condition (interrupt).

Test ID 14 DISC WRITE PROTECT Test

This is an operator intervention test that requires that a write-protected diskette is placed into the drive-under-test. An attempted WRITE SECTOR is performed and a Write-Protected Status should be generated. The same sector is then read to verify that the sector (data) was not written to disc.

Error Messages

RAM Tests

*** BANK x ADDRESS = xxxx DATA SB = xx IS = xx

*** RAM PARITY ERROR

SERIAL I/O Tests

*** NO SERIAL I/O RECEIVE DATA INTERRUPT
*** RECEIVE CHARACTER NOT AVAILABLE IN ALLOWED TIME
*** RECEIVE CHARACTER EARLY
*** TRANSMITTER BUFFER NOT EMPTY
*** RECEIVE DATA S/B xx IS xx
*** BAUD RATE xx (See Note)
*** SERIAL I/O PORT STATUS = xx

NOTE: Message displayed with any Serial I/O Test error.

60 Hz Timer Test

*** 60 Hz TIMER IS TOO SLOW
*** 60 Hz TIMER IS TOO FAST
*** NO 60 Hz TIMER INTERRUPT

DISC Tests

For each disc error, the controller command which produced the error will be displayed. Possible command messages are:

*** RESTORE COMMAND
*** SEEK COMMAND
*** STEP-IN COMMAND
*** STEP-OUT COMMAND
*** WRITE MULTIPLE SECTOR COMMAND
*** READ MULTIPLE SECTOR COMMAND
*** WRITE SECTOR COMMAND
*** READ SECTOR COMMAND

Additional possible error messages are;

*** DISC CONTROLLER STATUS = xx
*** TRACK UNDER TEST = xx (See Note 1)
*** TRACK REGISTER = xx
*** SECTOR xx DATA = xx SB = xx
*** SECTOR BYTE xx (See Note 2)
*** DATA WAS WRITTEN ON PROTECTED DISC

NOTE 1: Displayed with all Disc Errors.

NOTE 2: Displayed with all Disc (Data) Errors.

General Error Messages

*** UNEXPECTED DISC CONTROLER INTERRUPT
*** UNEXPECTED DMA INTERRUPT
*** UNEXPECTED GPIB INTERRUPT
*** INVALID TEST ID ***
*** INVALID COMMAND ***