XEROX®

TECHNICAL REFERENCE MANUAL

Xerox Professional Computer

[redacted]

9R80758

Reprint permission granted by Zilog, Inc. for Zilog Reprint on pages 49-108.

Reprint permission granted by Shugart Associates for SA 1403D Controller Reprint on pages 253-278.

Reprint permission granted by Western Digital for Western Digital Reprint on pages 279-302.

WARNING: This equipment has been certified to comply with the limits for a Class B computing device, pursuant to Subpart J of part 15 of FCC Rules. Only peripherals (computer input/output devices, terminals, printers, etc.) certified to comply with the Class B limits may be attached to this computer. Operating with non-certified peripherals is likely to result in interference to radio and TV reception.

WARNING: This equipment generates and uses radio frequency energy and if not installed and used properly, that is, in strict accordance with the manufacturer's instructions, may cause interference to radio and television reception. It has been type tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of part 15 of FCC Rules, which are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

Reorient the receiving antenna.
Relocate the computer with respect to the receiver.
Move the computer away from the receiver.
Plug the computer into a different outlet so that computer and receiver are on different branch circuits.

If necessary, the user should consult the dealer or an experienced radio/television technician for additional suggestions. The user may find the following booklet prepared by the Federal Communications Commission helpful. This booklet is available from the U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, D.C. 20402, STOCK NO. 004-000-00345-4.

"HOW TO IDENTIFY AND RESOLVE RADIO-TV PROBLEMS"

THIS PAGE INTENTIONALLY BLANK.

Table of Contents

Introduction

Introduction & Miscellaneous Information

- Hardware Description 2
- Configurations and Product Codes 3,4
- Functional Description 4,5
- Hardware Interface 5,6
- Operating Systems:
  - CP/M-80 6-9
  - CP/M-86 9,10
  - MS-DOS 10,11

Hardware

Miscellaneous Hardware Information

- Dimensions 13
- Electrical Requirements 13
- Operating Environment 13
- Disk Drive Storage 14
- Disk Drive Format 14
- Display Specification, Functional Description, and General Information 15-17
- System Bus Expansion Slot - Electrical & Environmental Requirements 18
- Hardware Interface - Connectors:
  - J1 (Disk), J2 (Keyboard), J3 (Printer), J4 (Comm) 19
  - J1 Illustration 20
  - J8 Illustration 21
  - J5, J6 (Power Supplies), J7 (Video Connector) 22
  - J8 (Parallel Connector) 23
  - Parallel Connector Illustration 24
  - J9 (COMM Port Strapping Options) 25
  - J9 (COMM Port Strapping Options) (Etch 2 CPU) 26
  - J13 (System Bus) 27-32
  - J12 (Disk Access) 33-37
  - J1 (Floppy Disk) 38
  - J1 (Rigid Disk) 39

Hardware continued

Theory of Operation

- CPU 41
- ROM and RAM Memory 41
- 6 - 8k ROM Memory 42
- 64k RAM 42
- Counter Timer Circuit 42
- Serial I/O Controller 42
- Serial I/O Ports 42,43
- Parallel I/O Controller 43
- Parallel Port 43
- CRT Controller 43,44
- CRT RAM Memory Allocation 45
- Scroll Register 46
- Speaker 47
- Disk Drive Daughter Board 47
- Rigid Disk Drive Caution 47
- System Bus 48
- Keyboard Interface 48

Zilog Reprint (SIO, PIO, CTC, & CPU Data sheets) 49-108

Schematics:

#1	Etch 1 CPU - Power Distribution	109
#2	Etch 1 CPU - CRT Controller	110
#3	Etch 1 CPU - CRT Controller	111
#4	Etch 1 CPU - U16, U26, U28, U32, U37-U41, U54, U57, U58	112
#5	Etch 1 CPU - U41, U54-U58, U66-U73	113
#6	Etch 1 CPU - RAM	114
#7	Etch 1 CPU - CTC	115
#8	Etch 1 CPU - GP, PIO, SIO	116
#9	Etch 2 CPU - Power Distribution	117
#10	Etch 2 CPU - CRT Controller	118
#11	Etch 2 CPU - CRT Controller	119
#12	Etch 2 CPU - U16, U26, U28, U32, U37-U41, U54, U57, U58	120
#13	Etch 2 CPU - U41, U54-U58, U66-U73	121
#14	Etch 2 CPU - RAM	122
#15	Etch 2 CPU - CTC, Keyboard, PIO	123
#16	Etch 2 CPU - GP, PIO, SIO	124

Hardware continued

#17	Floppy Controller - Power Distribution	125
#18	Floppy Controller	126
#19	Rigid Controller	127
#20	8086 Processor - Power Distribution & Connectors	128
#21	8086 Processor - U21-U24, U29-U32	129
#22	8086 Processor - U27, U33-U42	130
#23	8086 Processor - U1-U16, U18-U20, U25, U26, U28	131
#24	Memory Expansion PWB - 8086	132

Software

Device Initialization

- How the ROM works 133,134
  - Counter Timer Circuit 134
  - System PIO Controller 134
  - General Purpose PIO Controller 135
  - SIO Controller 135
  - Channels A & B Baud Rate Generator 136
- Monitor Commands - Summary Table 137
  - Display Memory, Modify Memory 138,139
  - Xtended Memory, Fill Memory 139
  - Copy Memory, Verify Memory Block, Go To 140,141
  - Input, Output 141,142
  - Load From Disk 142
  - Read Disk Sector, Write Disk Sector 143
  - Baud Rate, Typewriter, Host Terminal Mode 144
  - Host Terminal Options 145,146
  - Host Terminal Command Set 147,148
  - Protocol 148,149

Operating System Interface

- Accessing BIOS 151,152
- Logical/Physical Device Mapping 153-155
- I/O Port Assignments 156,157
- ROM Operating System Interface 158
- CRT Entries 158-160
- Execute Physical Driver Entries 161,162
- Printer Entries 163,164
- Communications Entries 165

Software continued

- Keyboard Entries 166
- IOBYTE Directed I/O 167-169
- Programmable Functions 170-176
  Sample Code Sequence 177-179
- Miscellaneous Functions - Cold, Warm, Getsel, Daytim, Config, Start Screen Print 177-182

CRT Control & Interface
- Modes of Operation 183
- Text Character Set 184
- Graphics Character Set 185
- Display Manipulation 186
- Control Sequences 186-188
- Escape Sequences 189-193

Peripherals

Keyboards
- ASCII 195-199
- Low Profile (LPK) 200-206

Disk Drive Specifications 207-212

Disk Parameter Headers 213-218

Disk Formats 219-226

20 & 40 CPS Printers
- 20 CPS Operating Switches 227-230
- 20 CPS Command Codes 231-233
- 20 CPS Command/Control/Hex Code Chart 234
- 20 CPS Specifications 235,236
- 20 CPS Cabling Requirements 236
- 40 CPS - HPRO5 Board 237
- 40 CPS Operating Switches 238-240
- 40 CPS Command Codes 242-244
- 40 CPS Command/Control/Hex Code Chart 245
- 40 CPS Electrical Interface 246-250

Peripherals continued

- 40 CPS Specifications 251-252
- 40 CPS Cabling Requirements 252

SA1403D Manual Reprint 253-278

Western Digital Reprint 279-302

Appendices

A. BIOS Entry Vector Table A1-A6

B. Monitor Entry Vector Table B1-B2

C. Documented System Storage & Structures
- Z80-A Mode 2 Interrupt Vectors C1, C2
- Keyboard Interface C1
- Logical to Physical Drive Mapping Table C2
- Physical Driver Address Table & Request Block C3
- Time-of-Day and Timer Variables C4

D. Program Samples
- How to make Monitor Calls with BASIC D1-D3
- Bank Program & Switching D4

E. ROM Listings E1-E110
- CBIOS E7-E12

F. Macros & Symbols F1-F24

G. Cold Start Loader G1-G6

H. BIOS Jump Table H1-H18

I. Banked Physical Driver I1-I12

J. Position-Encoded Keyboard Handler J1-J38

Appendices continued

K. Code Charts
- Baud Rates K1
- Decimal to ASCII to Hex K2

L. Additional References L1-L2

M. Contacts/reference: M1-M2
- Diablo
- Digital Research
- Microsoft
- Shugart

N. Abbreviations N1-N2

Introduction

The purpose of this manual is to provide technical reference material for the Xerox 820-II and 16/8 Professional Computers for programmers and engineers involved in hardware, software, and interface design. It is also intended for interested persons who have a desire to know how the Xerox 820-II and 16/8 operate and how to access their many features.

A list of the abbreviations and naming conventions used in this manual can be found in Appendix N.

SYSTEM OVERVIEW

The modular design of the 820-II and 16/8 systems enhance the flexibility provided by the operating systems. The combination of operations provided by the system gives it a flexibility that allows it to be tailored to the needs of each user.

Overview of Xerox Personal Computers

Xerox Personal Computers are comprised of four components: display/processor, disk drives, keyboard, and optional printers. Both the 820-II and 16/8 use the CP/M®-80 2.2 Operating System from Digital Research, Inc. Additionally, the 16/8 PC includes CP/M-86® 1.1and MS™-DOS Version 2.0 as standard operating systems.

HARDWARE DESCRIPTION

System Board

The system board uses a Zilog Z80-A®-based microprocessor operating on a 4 megahertz clock with 64k RAM and 8k ROM. It is a single-board computer and uses a daughter board to interface with the disk drives. The 820-II is equipped with three user-accessible I/O ports. Two of the ports are located on the back of the display/processor; the third is located on the CPU board. On the back are the printer and the communications ports (both RS232C). The port inside the display is a dual parallel port (most printers and other devices that follow a standard Centronics 36-pin interface can be successfully attached).

The 16/8 has all of the above features as well as an Intel 8086®-based microprocessor operating with a 4.772 megahertz clock. The 8086 is equipped with 128k of RAM which is expandable to 256k total by adding a 128k daughter board to the 8086 board.

The 820-II and 16/8 are capable of having up to 8k of read only memory (four 2k ROMs): the 820-II has 6k of this 8k occupied; the 16/8 uses the full 8K. The last 2k on the 16/8 is used for decoding the position-encoded Low Profile Keyboard. The firmware contained in the ROM is capable of doing such things as executing a one-sector loader from disk; i.e., loading CP/M, emulating a terminal, operating in typewriter mode, etc. The monitor also has other commands that are useful for debugging hardware and software. The mother board also contains a speaker as well as an expansion slot (used by the 16/8 for the 8086 board). There are two types of daughter boards: one interfaces the display/processor to floppy disks and the other interfaces the display/processor to a rigid disk controller.

Display

The display/processor houses the video display, the CPU mother board, the disk drive daughter board, and the 8086 processor board if so configured. The video display is a standard 24 line by 80 characters. It uses a 7 x 10 dot matrix for each character in all text modes and displays white characters on a black screen. For graphics characters, it uses a 4 x 4 pixel resolution. The display attributes can be changed to display either in blink, highlight/lowlight, inverse, or graphics characters.

820-II Display/processor for floppy disks	Product Code #U03
820-II Display/processor for rigid disk	Product Code #U05
16/8 Display/processor for floppy disks	Product Code #H69
16/8 Display/processor for rigid disk	Product Code #H70

Keyboards

The 820-II and 16/8 use either a standard 96-character ASCII or Low Profile keyboards. Both keyboards include additional keys to the right of the keyboard, a 10-key numeric key pad and a set of keys for software control of the cursor. The low profile keyboard also includes 12 function keys that can be software-enabled and other keys such as Accept, Delete, Next, Previous, Home, and Undo.

ASCII Keyboard	Product Code #X928
Low Profile Keyboard	Product Code #G25

Disk Drives

Five disk drive options are offered for the 820-II:

Dual $5\frac{1}{4}$" single-sided floppy disk drives	Product Code #X929
Dual $5\frac{1}{4}$" double-sided floppy disk drives	Product Code #T66
Dual 8" single-sided floppy disk drives	Product Code #X973
Dual 8" double-sided floppy disk drives	Product Code #F10
One 10mb rigid disk drive with an 8" double-sided disk drive	Product Code #U07

Three disk drive options are offered for the 16/8:

Dual 8" single sided floppy disk drives	Product Code #X973
Dual 8" double sided floppy disk drives	Product Code #F10
One 10mb rigid disk drive with an 8" double sided disk drive	Product Code #U07

Printers

40 CPS Printer and 20 CPS Printer

As their names imply, the printers have a printing speed of 20 and 40 characters per second (CPS) respectfully. Both printers have a wide range of print styles available. The 20 CPS Printer supports 10, 12, and 15 pitch as well as Proportional Spacing (PS), while the 40 CPS Printer supports either metal or plastic printwheels in 10, 12, 15, and PS. More detailed information on these printers can be found in the Printer section under Peripherals.

The standard RS232C printer connector and dual parallel port are available to interface with many types of serial and parallel printers.

40 CPS Printer	Product Code # D80
20 CPS Printer	Product Code # U01

FUNCTIONAL DESCRIPTION

The 820-II and 16/8 systems are a collection of four components working in unison -- the display, keyboard, disk drives, and printer. The computer itself is housed in the display.

System Monitor - ROM

The system monitor contained within the 8k ROM controls the essential functions of initializing and controlling all system input/output resources, and also provides a number of monitor commands that can be used to assist in programming.

Ports

Three ports are standard on the 820-II and 16/8: two serial ports located at the back of the display unit and an additional dual parallel port inside the display unit. These allow printers, communication devices, and other peripheral equipment to be interfaced with the system.

Operating Systems

The 820-II uses Digital Research's 2.2 CP/M-80 Operating System. The 16/8 can use Digital Research's 2.2 CP/M Operating System, as well as their CP/M-86 1.1 Operating System and Microsoft's MS-DOS 2.0 Operating System. These operating systems provide the user with a general environment for program construction, storage, and editing, along with assembly and program checkout facilities.

CP/M-80 operating system software as implemented on the 820-II and the 16/8 is logically divided into four parts:

ROSR	ROM Operating System Routines (hardware dependent)
BIOS	Basic I/O System (hardware dependent)*
BDOS	Basic Disk Operating System*
CCP	Console Command Processor*

*Disk resident portions of CP/M-80

ROSR provides code in ROM that can be executed without the presence of the CP/M system disk and provides the primitive operations necessary to access the disk drives and to interface with peripherals.

BIOS provides the interface between BDOS and ROSR.

BDOS provides disk management by controlling one or more disk drives containing independent file directories.

CCP provides symbolic interface between the user's console and the remainder of the CP/M system.

HARDWARE INTERFACE

The 820-II and 16/8 are equipped with six input/output connectors. Four are on the back of the display unit and two are inside the display.

Disk Drive

Used for connection of either the 8" or the 5¼" Dual Floppy Drives, or the 8" Rigid Disk Drive. This is determined by the type of disk daughter board installed in the display processor.

Keyboard

Used for connection of either the ASCII or Low Profile keyboard.

Printer

A serial printer can be attached to this RS-232-C connector.

COMM

COMM is a second RS-232-C connector and can be used for a modem.

Parallel Port
A dual parallel port inside the display cabinet is also provided.

Expansion Slot
The expansion slot inside the display cabinet provides all of the Z80-A microprocessor control signals for connection to custom devices for future expansion. This slot is used for the 8086 co-processor if you have a 16/8.

CP/M-80

The CP/M-80 2.2-C disk for the 820-II contains the standard Digital Research software development and checkout programs. Xerox issues additional utility programs that are unique to the 820-II. A description of each program is listed below:

Digital Research Files

ASM.COM	The Assembler allows you to create a program which can be read and executed by the 820-II.
DDT.COM	The Dynamic Debugging Tool is used to debug a Z80-A assembly language program.
DUMP.COM	Allows binary command files that are not displayed on screen to be displayed showing the hexadecimal value for each byte.
ED.COM	A line-oriented screen editor.
L80.COM	Reads an .REL file created with the Macro-80 Assembler Program and outputs a command file.
LOAD.COM	Reads a .HEX file and creates a command file.
M80.COM	Converts a program written in Assembly Language to a relocatable (.REL) file and (optionally) a printer listing file (.PRN).
MOVCPM.COM	Lets you modify and move the CP/M system image to allocate a specific lesser memory size.
PIP.COM	Allows you to selectively copy a file or files from one disk to another or on the same disk.
STAT.COM	The status utility is a frequently-used transient command for all system housekeeping; i.e., checking the amount of space available on a disk.
SUBMIT.COM	Used to submit a file of commands for batch processing.

SYSGEN.COM	Used to generate a CP/M-80 system image and copy the operating system to another disk.
XSUB.COM	Same as Submit.com, but has the facility to include line input to programs as well as the console command processor.

Xerox Files

BACKUP.COM — A multi-option utility that allows you to archive and retrieve files, delete files, list directories of any drive, and to verify data integrity of a floppy or rigid disk.

CONFIGUR.COM — Using Configur.com, you can select seven different options:

1. Record Restart Command - lets you enter a one-line command which will automatically load a program. For example, you could enter DIR as the restart command and every time you boot the system, it will automatically display the directory for you. Or you could enter the name of your application software package and it would automatically load that application package for you. This command is recorded on the disk and you can have a different one for each disk.
2. Select Printer Port Options - allows you to determine printer protocol. This option allows configuration for alternate printers without modifying the BIOS.
3. Select Communications Port Options - a convenient method for setting up the communications port on the 820-II or 16/8; that is, baud rate, protocol, stop bits, etc.
4. Select I/O Device Assignments - lets you select alternative input/output device assignments; i.e., set up the system so that everything displayed on the screen automatically prints on the printer.
5. Select Keyboard Data Format - lets you choose 7-bit or 8-bit mode for the keyboard.
6. Select Screen Attributes - includes blink, inverse video, highlight/lowlight, and graphics modes.

7. If you have a floppy disk system, Select Floppy Disk Head Step Rate will appear as selection 7. If you have a rigid disk system, Configure Rigid Disk will appear (program must be loaded from floppy or the first partition of the rigid).
   a. Select Floppy Disk Head Step Rate - lets you adjust the floppy head step rate for optimum performance.
   b. Configure Rigid Disk - lets you divide the eight megabyte rigid disk into sections (e.g., 4 Mb, 2 Mb, 1 Mb, 1 Mb).

COPY.COM	Makes an exact copy of a disk, track for track.
FMT.COM	Allows you to format (initialize) a rigid disk. Verification of the rigid disk is performed using the Backup.com utility.
HELP.COM	A guide for CP/M-80 users that contains basic information about CP/M-80 commands; also cross-references to additional information in the CP/M-80 reference manual, Reorder #9R80448.
INIT.COM	Prepares new (or used) disks for storing information. It will also alert the user to any flawed sectors on the disk.
KILLESC.COM	Turns off the <CTRL> + <ESC> feature to enable use of <CTRL> + <ESC> for other purposes; for example, setting margins and tabs on a 40 CPS printer uses a <CTRL> + <ESC> sequence.
SET.COM	A convenient method to temporarily change communication and printer port options in RAM.
SWAP.COM	A utility that allows the user to swap drive names. For example, "A" and "E" for a rigid disk drive. By designating an alternate drive as the "A" drive, you can load software directly from that drive. Many CP/M-80 application packages have been written to be executed from the "A" disk drive only. Using Swap.com allows you to place your application software on any disk drive and load.
TIME.COM	Displays the time and date on screen. Since there is no battery backup, however, you must re-enter the time and date each time you reload the system.
WHATSA.COM	This utility lists the logical and physical names for each disk drive, as well as the density, number of

sides, and types of disks logged into the system, (e.g., double density, single-sided 8" floppy).

CP/M-86

The CP/M-80 2.2 and CP/M-86 1.1-F disks for the 16/8 contain the standard Digital Research software development and checkout programs. These disks contain the same files as described in the CP/M-80 section as well as the following files.

Digital Research Files

ASM86.CMD	The Assembler allows you to create a program which can be read and executed by the 8086.
DDT86.CMD	The Dynamic Debugging Tool is used to debug a 8086 assembly language program.
ED.CMD	A line-oriented screen editor.
GENCMD.CMD	Uses the hex output of ASM-86 and other language processors to produce a .CMD file.
GENCMD.COM	Uses the hex output of ASM-86 and other language processors to produce a .COM file.
GENDEF.CMD	Reads a 16-bit file containing the disk definition statements, and produces a 16-bit output file containing assembly language statements which define the tables necessary to support a particular drive configuration.
GENDEF.COM	Reads a 16-bit file containing the disk definition statements, and produces an 8-bit output file containing assembly language statements which define the tables necessary to support a particular drive configuration.
HELP.CMD	Provides summarized information for all of the CP/M-86 commands described in the Digital Research Users manual.
LMCMD.CMD	Operates in exactly the same manner as Gencmd.cmd, except Lmcmd also accepts an Intel L-module file as input.
LMCMD.COM	Operates in exactly the same manner as Gencmd.com except Lmcmd also accepts an Intel L-module file as input.
PIP.CMD	Allows you to selectively copy a file or files from one disk to another or on the same disk.

STAT.CMD	The status utility is a frequently-used transient command for all system housekeeping, i.e., checking the amount of space available on a disk.
SUBMIT.CMD	Used to submit a file of commands for batch processing.
TOD.CMD	Time of day.

Xerox Files

CPM86.COM	Used by Load86.com to boot the 8086.
86CON.COM	Switches from Z80-A console to the 8086 console.
GOBACK.CMD	Switches from 8086 console to the Z80-A console.
LOAD86.COM	Loads the 8086 for concurrent processing.
REBOOT.COM	From the concurrent mode, reboots the system as a Z80-A standalone.
SOFTKEYS.COM	Used to set up the 10-key pad with programmable functions (<CTRL> + one of the 10-key pad keys).

MS-DOS

The MS-DOS 2.0 disk for the 16/8 contains the standard Microsoft software development and checkout programs.

Microsoft Files

ANSI.SYS	Allows programs that use the standard ANSI driver to be executed.
COMMAND.COM	This is the MS-DOS command processor. It is recommended that this file be placed on every application program disk.
CONFIG.SYS	Configures system at boot.
CHKDSK.COM	Checks disk.
CREF.EXE	Assists in debugging assembly language programs.
DEBUG.COM	Debugger supplied with MS-DOS.
DISKCOPY.COM	Copies a disk.
EDLIN.COM	Line-oriented screen editor.
EXE2BIN.EXE	Converts .EXE files to binary format.
FC.EXE	Compares two files for similarity.
FIND.EXE	Finds a string in a list of files or standard input.
FORMAT.COM	Formats an 8" floppy or a rigid disk.
LINK.EXE	Linker.
MORE.COM	Used to display text in 23-line segments.

MASM.EXE	Macro Assembler for MS-DOS.
PRINT.COM	Print spooler.
RDCPM.COM	Reads a CP/M-80 file and converts data to MS-DOS-readable file.
RECOVER.COM	Recovers bad or damaged disks.
SORT.EXE	Used to sort text.

Xerox Files

SAMPLE.TXT	Provided to assist going through MS-DOS Handbook.

Notes

Specifications
This section details the following specifications: dimensions, electrical requirements, operating environment, and disk drive capacities of the Xerox 820-II and 16/8 PCs.

Dimensions

Equipment	Height	Depth	Width	Weight
820-II-16/8 Display	12.20"	14.75"	15.00"	30 lbs.
ASCII keyboard	3.75"	9.50"	20.00"	10 lbs.
Low profile keyboard	1.60"	8.25"	19.90"	5 lbs.
5¼" Floppy disk drives	7.00"	10.20"	7.00"	10 lbs.
8" Floppy disk drives	10.50"	17.50"	14.50"	48 lbs.
8" Rigid disk drive	10.50"	17.50"	15.50"	54 lbs.
40 CPS printer	10.00"	17.50"	15.50"	56 lbs.
20 CPS printer	9.25"	17.50"	24.00"	45 lbs.

Electrical Requirements
All Xerox products listed below require voltage of 115 VAC, a frequency of 60 Hz, and a two-pole, three-wire grounded duplex receptical.

Equipment	Current
820-II-16/8 Display	1.1 Amps
8" Floppy disk drives	2.0 Amps
8" Rigid disk drive	2.2 Amps
40 CPS printer	2.0 Amps
20 CPS printer	1.0 Amp

Operating Environment
All Xerox equipment is tested to perform between 50 and 90 degrees Fahrenheit with a relative humidity factor between 20% and 80%.

Disk Drive Storage	Unformatted	Formatted	Usable
5¼" SS/SD	125 k	90 k	81 k
5¼" SS/DD	250 k	168 k	155 k
5¼" DS/SD	250 k	180 k	172 k
5¼" DS/DD	500 k	338 k	322 k
8" SS/SD	400 k	250 k	241 k
8" SS/DD	800 k	497 k	482 k
8" DS/SD	800 k	500 k	490 k
8" DS/DD	1.6 Mb	997 k	980 k
8" Rigid (DS/DD)	10.67 Mb	8.4 Mb	8.192 Mb

Disk Drive Format

Equipment	Tracks	Sectors	Bytes per Sector	Number of Heads
5¼" SS/SD	40	18	128	1
5¼" SS/DD*	40	17	256	1
5¼" DS/SD	80	18	128	2
5¼" DS/DD*	80	17	256	2
8" SS/SS	77	26	128	1
8" SS/DD**	77	26	256	1
8" DS/SD	154	26	128	2
8" DS/DD**	154	26	256	2
8" Rigid DS/DD	1,024	32	256	4

*Track 0 of 5¼" double density disks has 18 sectors of 128 bytes.
**Track 0 of 8" double density disks has 26 sectors of 128 bytes.

For more specific information on disk formats, see the Disk Drive Specifications section.

DISPLAY SPECIFICATION

SIZE:	12 inch, landscape mode	
TYPE:	Aluminized	P4
	Fluorescence	White (W)
	Phosphorescence	White (W)
	Persistence	Short
RESOLUTION:	• 240 active line raster adjusted to 8.5 x 5.3 inch usable area • Brightness level 30 (± 2) foot-lamberts • Resolution at centers (within 1" diameter circle) -100 lines/inch minimum	
CHARACTER CELL:	7x10	
BUSINESS GRAPHICS:	4x4 Pixel Resolution	
CHARACTER SET:	4 sets of 128:	(1 U.S. font, 1 Graphics font) (1 U.S. font, Inverse Video font)
CHARACTER LINES:	24	
CHARACTERS/LINE:	80	
VOLTAGE:	+ 12 (± 5.0%) VDC at 2.0 A DC maximum	
RIPPLE:	50 MV P-P synchronous or nonsynchronous with refresh or power frequency.	
VIDEO BIT RATE:	10.694 MBPS	(93.51 nanoseconds)
BITS/HORZ LINE:	560	
HORZ SYNC PULSE:	126	(11.78 microseconds)
TOTAL BITS/LINE:	686	
HORZ RATE:	15.59 KHz	(64.14 microseconds)
LINES/FIELD:	240	
VERT BLANKING LINES:	20	
VERT SYNC PULSE:	20(1.28 milliseconds)	
VERT RETRACE (lines):	8 TYP	
TOTAL LINES/FIELD:	260	
FIELD RATE:	59.95 Hz	(16.68 milliseconds)
REFRESH RATE:	61 Hz	
VIDEO RATE:	15 MHz	

FUNCTIONAL DESCRIPTION, XEROX DISPLAY

The display has the following functional characteristics:

- 24 line display
- 80 characters per line
- 7x10 dot matrix per character
- White characters on black
- Software-selectible character attributes
  - Inverse Video
  - Blink
  - Low Intensity
  - Graphics with 4 x 4 pixel resolution
- Brightness adjust

DISPLAY CONTROLLER

The Display Controller is based on displaying characters within a 7x10 cell (7 dots horizontally by 10 scan lines vertically). To guarantee spaces between characters, one dot on each side of the cell is blanked by hardware. Also, to guarantee spaces between character lines, the top two scan lines are blanked by hardware. This gives an actual active character size of 5 dots horizontally by 8 scan lines vertically.

For Business Graphics, the hardware is configured to eliminate the automatic blanking and allow continuous lines both horizontally and vertically. However, the Display Controller is still based on displaying a character within a 7 x 10 cell. The controller design and available refresh memory allows one byte per character. The maximum number of unique characters that can be defined by any 8 bits is 256. Since the standard text font set contains 128 characters, the limit on unique characters for graphics that can be displayed together with text is 128.

The character set for Business Graphics divides the cell into blocks of 4 dots horizontally by 4 scan lines vertically. Since the total number of scan lines per character is 10, the character set actually consists of two sub-sets of 4-4-2 and 2-4-4.

Each subset divides the cell into 6 parts requiring 64 possible combinations or unique characters. Therefore, the total number of unique characters for the complete graphics set is 128. With this character set, any combination of adjacent 4 x 4 blocks can be chosen. Also, at the character cell boundary, the 4 x 4 blocks can be set vertically by 2 scan lines. Since the total number of horizontal dots per cell is 7, there will be an overlap of one horizontal dot in the center of the cell for diagonal blocks within the cell.

It should be also noted that for the standard text font containing 128 unique characters defined by 7 bits, the eighth bit is used to set the attribute function. For Business Graphics, since both text characters and graphic characters can be displayed simultaneously, it requires all 8 bits to define the character. Consequently, display attributes are not available in graphics mode.

SYSTEM BUS EXPANSION SLOT

ELECTRICAL

The DC system power available at the expansion slot is as follows:

		$5\frac{1}{4}$" system	8" or Rigid system
PIN 50	+ 5V DC	1.2A	2.1 A
PIN 45	+ 12V DC #1	0.3A	1.75 A

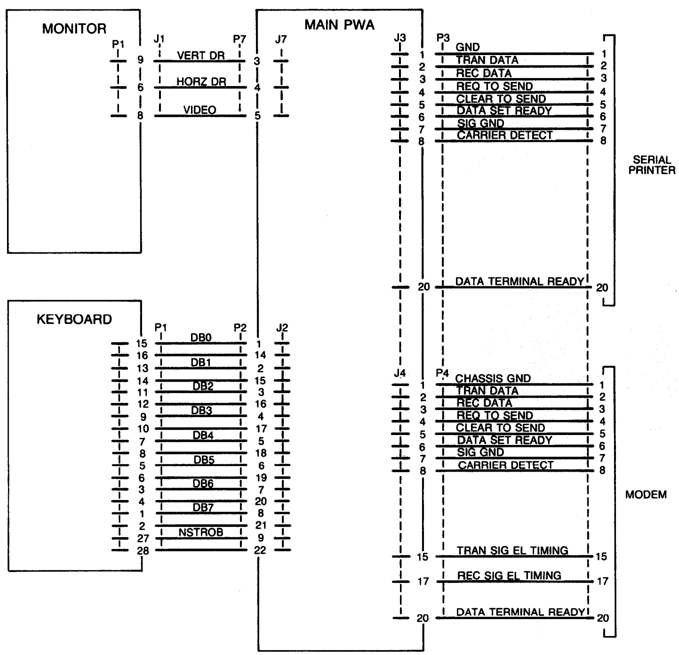
ENVIRONMENTAL

The following temperature, humidity and altitude environmental requirements are specified:

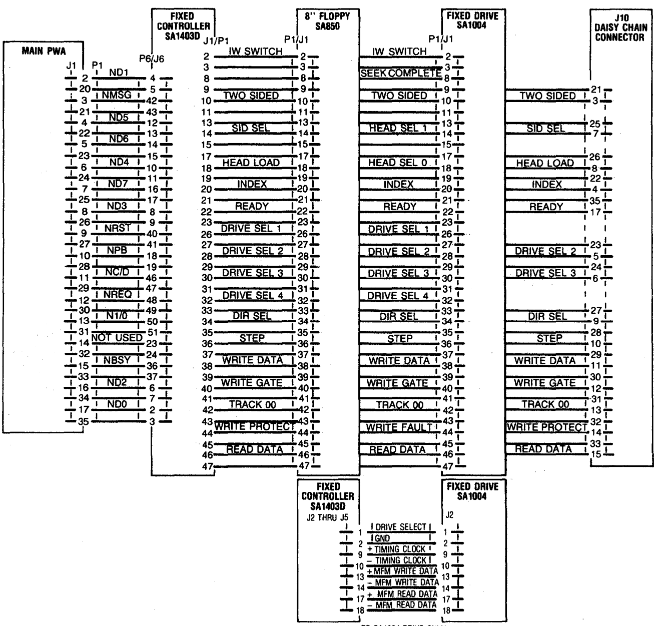
	Temp. (°Celsius)	Rel.Hum. (%)	Altitude (miles)
Operating	10 to 32	20 to 80	1830
Non-operating	-77 to 66	15 to 90	7620

Any optional or additional electronic assembly using the expansion slot must be capable of performing to design specification when the host is subjected to the environmental range, above. Furthermore, the presence of such an assembly in the expansion slot must not degrade performance with regard to the above environmental requirements.

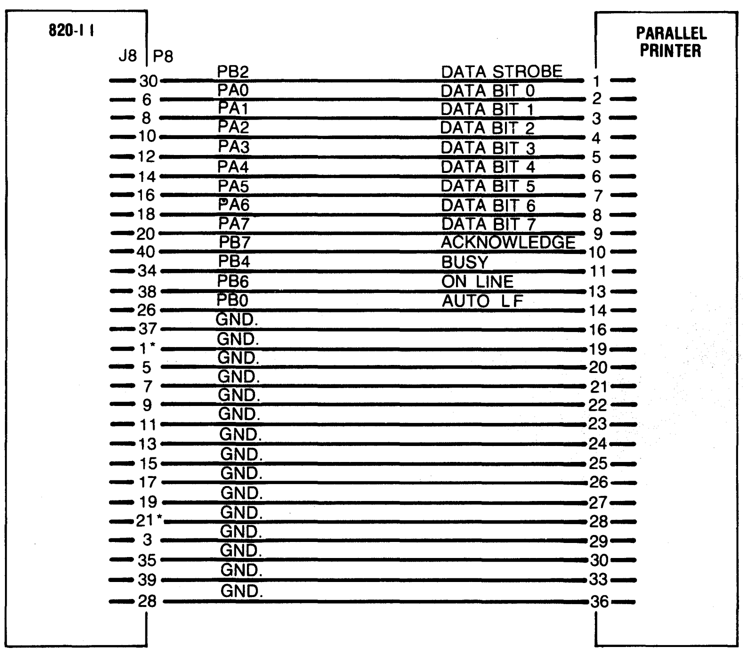
HARDWARE INTERFACE



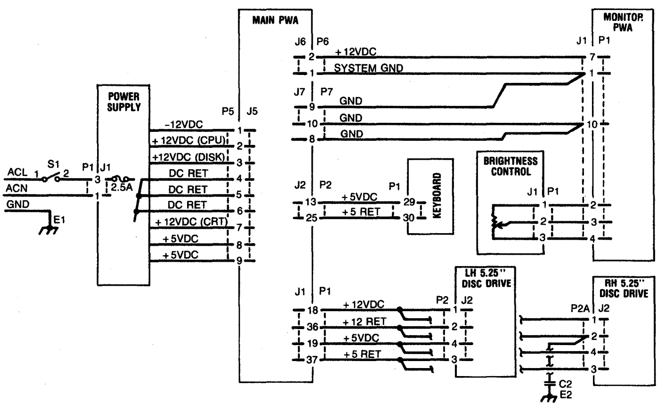
J1 illustration



J8 illustration



Power Supplies, and Video Connectors



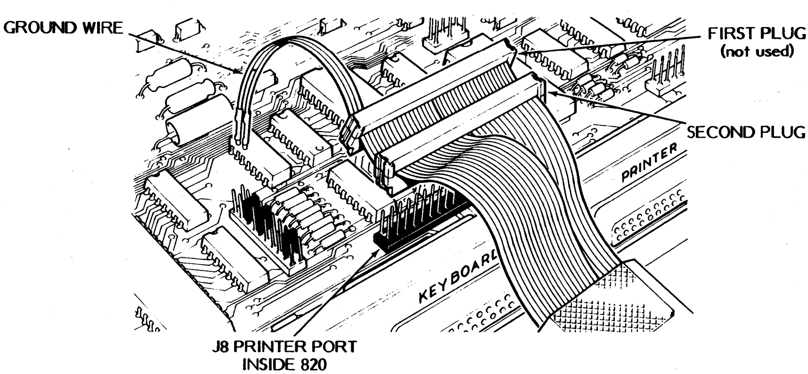
Parallel Port Connector (J8)

```
39                                              1
      o o o o o o o o o o o o o o o o o o o o
      o o o o o o o o o o o o o o o o o o o o
40                                              2
```

J8

Pin	Value
2	Port A Strobe
4	Port A Ready
6	Port A Bit 0
8	Port A Bit 1
10	Port A Bit 2
12	Port A Bit 3
14	Port A Bit 4
16	Port A Bit 5
18	Port A Bit 6
20	Port A Bit 7
22	Port B Ready
24	Port B Strobe
26	Port B Bit 0
28	Port B Bit 1
30	Port B Bit 2
32	Port B Bit 3
34	Port B Bit 4
36	Port B Bit 5
38	Port B Bit 6
40	Port B Bit 7
Odd # Pins	Ground

Parallel connector picture



COMM Port Strapping Options

```
39                                        1
     oooooooooooooooooooooo
     oooooooooooooooooooooo
40                                        2
```

J9

Pin	Value
5--6	(M) TXD to Pin 3
7--8	(T) TXD to Pin 2
9--10	(M) RXD from Pin 2
11--12	(T) RXD from Pin 3
13--14	(M) RTS to Pin 5
15--16	(T) RTS to Pin 4
17--18	(M) CTS from Pin 4
19--20	(T) CTS from Pin 5
21--22	(M) DTR to Pin 8
23--24	(T) DTR to Pin 20
25--26	(M) DCD from Pin 20
27--28	(T) DCD from Pin 8
29--30	Clock supplied to Modem as RX Clock
31--32	Clock supplied to SIO with RX Clock
33--34	Modem supplies SIO with RX Clock
35--36	Clock supplied to SIO with TX Clock
37--38	Modem supplies SIO with TX Clock
39--40	Clock supplied to Modem with TX Clock

The filled-in pins indicate the options as they are jumpered on an 820-II or 16/8.

Note: (M) indicates modem (data communications equipment) function. (T) indicates terminal data equipment) function. For instance, exercising the (M) strap option will allow communication with a modem; exercising the (T) strap option will allow communication with a terminal. The above shows factory settings for (T).

J9 (Etch 2 CPU)

```
29                                             1
        o o o o o o o o o o o o o o o
        o o o o o o o o o o o o o o o
30                                             2
```

Pin	Value
3--4	
5--6	(M) TXD to Pin 3
7--8	(T) TXD to Pin 2
9--10	(M) RXD from Pin 2
11--12	(T) RXD from Pin 3
13--14	(M) RTS to Pin 5
15--16	(T) RTS to Pin 4
17--18	(M) CTS from Pin 4
19--20	(T) CTS from Pin 5
21--22	(M) DTR to Pin 8
23--24	(T) DTR to Pin 20
25--26	(M) DCD from Pin 20
27--28	(T) DCD from Pin 8
29--30	Clock supplied to Modem as RX Clock

The filled-in pins indicate the options as they are jumpered on an 820-II or 16/8.

Note: To change from ASYNC to SYNC on the Etch 2 CPU requires a modification to the operating system rather than moving jumpers.

Note: (M) indicates modem (data communications equipment) function. (T) indicates terminal data equipment) function. For instance, exercising the (M) strap option will allow communication with a modem; exercising the (T) strap option will allow communication with a terminal.
The above shows factory settings for (T).

System Bus Connector

J13

D1	1	2	/RD
D0	3	4	/MEMRQ
D7	5	6	/IORQ
D2	7	8	/WR
D6	9	10	/REFRESH
D5	11	12	/M1
D3	13	14	A0
D4	15	16	A1
SYSRESET	17	18	A2
A4	19	20	A3
A6	21	22	A5
A15	23	24	A7
A13	25	26	A14
A12	27	28	A10
A9	29	30	A11
A8	31	32	/BUSRQ
WAITRQ	33	34	/BUSAK
PCI	35	36	
/INTRQ	37	38	
/HALT	39	40	/CLOCK
SPKR	41	42	/MEM8
	43	44	/MEM4
+ 12V	45	46	
+ 12V	47	48	GND
GND	49	50	+ 5V

Symbol	Pin #	Pin Name	Meaning
D0	3	Data bus	Data Bus (Tri-state, input/output, active high) constitutes an 8-bit bi-directional data exchange with memory and I/O devices.
D1	1	Data bus	
D2	7	Data bus	
D3	13	Data bus	
D4	15	Data bus	
D5	11	Data bus	
D6	9	Data bus	
D7	5	Data bus	
A0	14	Address bus	Address Bus (Tri-state, output, active high) makes up a 16-bit address for up to 65k bytes of memory for I/O devices data exchange. I/O addressing uses the lower 8 bits for direct selection of up to 256 output ports. A0 is the least significant address bit. During refresh time, the lower 7 bits contain a valid refresh address for dynamic memories.
A1	16	Address bus	
A2	18	Address bus	
A3	20	Address bus	
A4	19	Address bus	
A5	22	Address bus	
A6	21	Address bus	
A7	24	Address bus	
A8	31	Address bus	
A9	29	Address bus	
A10	28	Address bus	
A11	30	Address bus	
A12	27	Address bus	
A13	25	Address bus	
A14	26	Address bus	
A15	23	Address bus	
/WR	8	Write	Write (Tri-state, output, active low) indicates that the CPU data bus holds valid data to be stored in the addressed memory or I/O device.
/RD	2	Read	Read (Tri-state, output, active high) indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.
/IORQ	6	I/O Request	Input/Output Request (Tri-state, output, active low) signal indicates that the lower half of

			the address bus holds a valid I/O address for an I/O read or write operation. This signal is also generated with a "/M1" signal when an interrupt is being acknowledged to indicate that an interrupt response vector can be placed on the data bus. Interrupt Acknowledge operations occur during "/M1" time, while I/O operations never óccur during "/M1" time.
/HALT	39	Halt	Halt (Output, active low) signal indicates that the CPU has executed a Halt Software instruction and is awaiting either a non-maskable or maskable interrupt before operation can resume.
/MEMRQ	4	Memory Request	Memory Request (Tri-state, output, active low) signal indicates that the address bus holds a valid address for a memory read or memory write operation.
/REFRESH	10	Refresh	Refresh (Tri-state, output, active low) indicates that the lower 7 bits of the address contain a refresh address for dynamic memories and the "/MEMRQ" signal should be used to perform a refresh cycle for all dynamic RAMs in the system. During the refresh cycle "A7" is a logic zero and the upper 8 bits of the address bus contain the "I" register.
/M1	12	Machine Cycle One	Machine Cycle One (Tri-state, output, active low) indicates that the current machine cycle is in the op-code fetch cycle of an instruction. Note

			that during the execution of two-byte op-codes, "/M1" will be generated as each op-code is fetched. These two-byte op-codes always begin with a CB, DD, ED, or FD. "/M1" also occurs with "/IORQ" to indicate an interrupt acknowledge cycle.
/BUSAK	34	Bus Acknowledge	Bus Acknowledge (Output, active low) is used to indicate to the requesting device that the CPU address bus, data bus, and control bus signals have been set to their high impedance states and the external device can now control the bus.
/BUSRQ	32	Bus Request	Bus Request (Input, active low) signal is used to request the CPU address bus, data bus, and control signal bus to go to a high impedance state so that other devices can control those buses. When "/BUSRQ" is activated, the CPU will set these buses to a high impedance state as soon as the current CPU machine cycle is finished and the "/BUSAK" signal is activated.
/INTRQ	37	Interrupt Request	Interrupt Request (Input, active low) signal is generated by I/O devices. A request will be honored at the end of the current instruction if the internal software controlled interrupt enable flip flop (IFF) is enabled and if the "/BUSRQ" signal is not active.
/WAITRQ	33	Wait Request	Wait Request (Input, active low) indicates to the CPU that the addressed memory or I/O device is not ready for a data transfer. The CPU continues to enter wait states

			for as long as this signal is active. This signal allows memory or I/O devices of any speed to be synchronized to the CPU. Use of this signal postpones refresh as long as it is held active.
/SYSREST	17	System Reset	System Reset (Output, active low) indicates that a reset has been generated either from push button reset or the power on reset circuit. The system reset will occur only once per reset and will be approximately 10 microseconds in duration.
/CLOCK	40	Processor Clock	Processor Clock (Output, active low) is a single-phase system clock of 4 MHz.
PCI	35	Priority Chain In	Priority Chain In (Input, active high) is used to form a priority-interrupt daisy chain when more than one interrupt-driven device is being used. A high level on this pin indicates that no other devices of higher priority are being serviced by a CPU interrupt service routine.
/MEM4	44	Memory Expansion	Memory Expansion (Output, active low) signal is low during "/MEMRQ" for a block of addresses from "4000 thru 7FFF" if the Bank Switch is set for the ROM side of memory.
/MEM8	42	Memory Expansion	Memory Expansion (Output, active low) signal is low during "/MEMRQ" for a block of addresses from "8000 thru BFFF" if the Bank Switch is set for the ROM side of memory.
/SPKR	41	Speaker	Speaker pin provides access to the speaker on the CPU Board. This pin is connected to the open collector output of the speaker

			driver (75451). This output is normally connected thru the speaker and parallel 100 ohm resistor to a + 12 VDC, but can be disconnected by jumper option.
+ 5VDC	50	DC Power	+ 5VDC system power.
GND	49,48	Ground	Ground-System is signal ground and DC return.
+ 12VDC	47,45	DC Power	+ 12VDC system power.
	36	Not Used	
	38	Not Used	
	43	Not Used	
	46	Not Used	

Disk Access Connector

J12

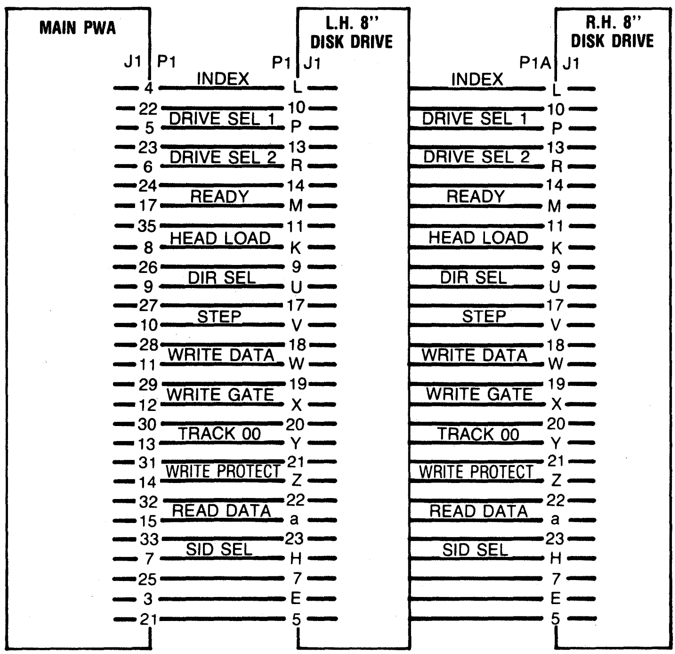
D1	1	2	/RD
D0	3	4	/MREQ
D7	5	6	/IORQ
D2	7	8	/WR
D6	9	10	/BUSAK
D5	11	12	/M1
D3	13	14	A0
D4	15	16	A1
RST	17	18	A2
A4	19	20	A3
A6	21	22	A5
A15	23	24	A7
A13	25	26	A14
A12	27	28	A10
A9	29	30	A11
A8	31	32	/BUSRQ
16 MHz Clock	33	34	/BUSAK1
/HALT	35	36	/1797CS
INT	37	38	PRIO
PP5	39	40	4 MHz Clock
PP2	41	42	PP4
PP1	43	44	PP0
+ 12V	45	46	NM1
GND	47	48	GND
DSKWAT	49	50	+ 5V
	51	52	+ 5V
J1-19	53	54	J1-10
J1-18	55	56	J1-9
J1-17	57	58	J1-8
J1-16	59	60	J1-2
J1-15	61	62	J1-3
J1-14	63	64	J1-4
J1-13	65	66	J1-5
J1-12	67	68	J1-6
J1-11	69	70	J1-7
	71	72	

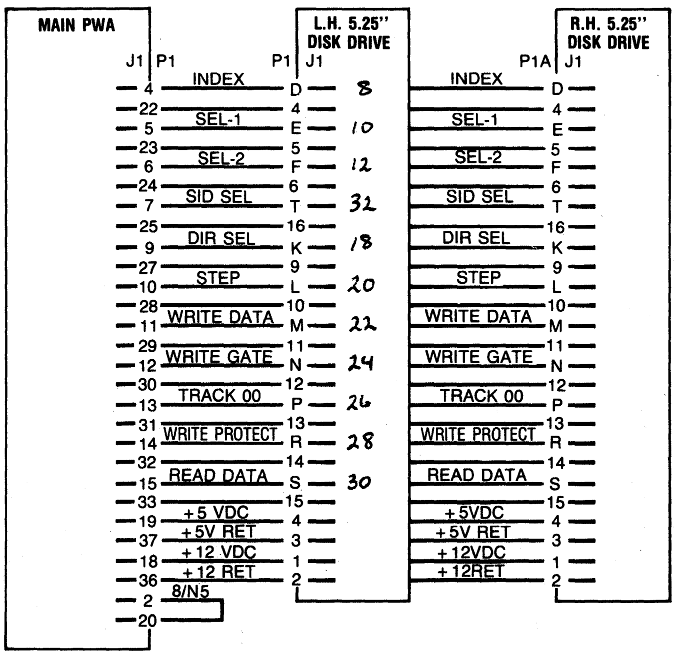
Symbol	Pin #	Pin Name	Meaning
D0	3	Data bus	Data Bus (Tri-state, input/output,
D1	1	Data bus	active high) constitutes an 8-bit
D2	7	Data bus	bi-directional data exchange with
D3	13	Data bus	memory and I/O devices.
D4	15	Data bus	
D5	11	Data bus	
D6	9	Data bus	
D7	5	Data bus	
A0	14	Address Bus	Address bus A0-A15
A1	16	Address Bus	provides addresses for 65k
A2	18	Address Bus	bytes of memory. Bit A0 and
A3	20	Address Bus	A1 while under /RD and /WR
A4	19	Address Bus	control select the register
A5	22	Address Bus	to receive transfer of data
A6	21	Address Bus	on D0-D7:
A7	24	Address Bus	A1 A0 /RD /WR
A8	31	Address Bus	0 0 Status REG Command REG
A9	29	Address Bus	0 1 Track REG Track REG
2A10	8	Address Bus	1 0 Sector REG Sector REG
A11	30	Address Bus	1 1 Data REG Data REG
2A12	7	Address Bus	A5 while under /RD and /WR
A13	25	Address Bus	Control Select Density:
A14	26	Address Bus	0 = double density
A15	23	Address Bus	1 = single density
PP0	44	SYS-PIO Port A	Port A bit 0
PP1	43	SYS-PIO Port A	Port A bit 1
PP2	41	SYS-PIO Port A	Port A bit 2
PP4	42	SYS-PIO Port A	Port A bit 4
PP5	39	SYS-PIO Port A	Port A bit 5
PRIO	38	SYS-PIO	
/DSKWAT	49	Disk Wait	Generates Wait signal to CPU.
/RD	2	Read	Controls input on the data registers D0-D7.
/MREQ	4	Memory Request	/MREQ indicates that the address bus holds a valid address for a memory read or memory write operation.

/IORQ	6	I/O Request	/IORQ indicates that the lower half of the address bus holds a valid I/O address for an I/O read or write operation. /IORQ is also generated concurrently with /M1 during an interrupt acknowledge cycle to indicate that an interrupt response vector can be placed on the data bus.
/WR	8	Write	Controls output on the data registers D0-D7.
/BUSACK	10	Bus Acknowledge	/BUSACk indicates to the requesting device that the CPU address bus, data bus, and control signals /MREQ, /IORQ, /RD, /WR have entered the high impedance states. The external circuitry can now control these lines.
/M1	12	Machine Cycle 1	/M1, together with /MREQ indicates that the current machine cycle is the op-code fetch cycle of an instruction execution. /M1, together with /IORQ, indicates an interrupt acknowledge cycle.
/BUSREQ	32	Bus Request	/BUSREQ has the highest priority and is always recognized at the end of the current machine cycle. /BUSREQ forces the CPU address bus, data bus, and control signals /MREQ, /IORQ, /RD, and /WR to go to a high impedance state so that other devices can control these lines.
/BUSAK1	34	Bus Acknowledge	/BUSAK1 is daisy-chained Bus Acknowledge output which indicates to the requesting device that the CPU address bus, data bus, and control signals /MREQ, /IORQ, /RD, /WR have entered the high impedance states. The

			external circuitry can now control these lines.
/1797CS	36	Chip Select	/1797CS logic low selects the Floppy Disk Controller chip and enables computer communication with the device.
INT	37	Interrupt Request	INT is generated by I/O devices. The CPU honors a request at the end of the current instruction if the internal software controlled interrupt enable flip-flop (IFF) is enabled.
NMI	46	Non-Maskable Interrupt	NMI is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop and automatically forces the CPU to restart at location 0066h.
/HALT	35	Halt	/HALT indicates that the CPU has executed a Halt instruction and is awaiting either a non-maskable or a maskable interrupt (with the mask enabled) before operation can resume. While halted, the CPU executes NOPs to maintain memory refresh.
16MHz	33	Clock	16 MHz clock.
CLK	40	Clock	4 MHz clock.
J1-2	60	Device I/O Interface	All interface lines use negative logic.
J1-3	62	Device I/O Interface	
J1-4	64	Device I/O Interface	
J1-5	66	Device I/O Interface	
J1-6	68	Device I/O Interface	
J1-7	70	Device I/O Interface	
J1-8	58	Device I/O Interface	
J1-9	56	Device I/O Interface	
J1-10	54	Device I/O Interface	
J1-11	69	Device I/O Interface	
J1-12	67	Device I/O Interface	
J1-13	65	Device I/O Interface	
J1-14	63	Device I/O Interface	

J1-15	61	Device I/O Interface	
J1-16	59	Device I/O Interface	
J1-17	57	Device I/O Interface	
J1-18	55	Device I/O Interface	
J1-19	53	Device I/O Interface	
+ 12VDC	45	DC Voltage	+ 12 Volts DC
GND	47	Ground	Ground and DC Return
GND	48	Ground	Ground and DC Return
+ 5VDC	50	DC Voltage	+ 5 Volts DC
+ 5VDC	52	DC Voltage	+ 5 Volts DC
RST	17	Reset	Reset indicates that a System Reset has been generated either from push button reset or power on reset.
	51	Not Used	
	71	Not Used	
	72	Not Used	


MAIN PWA
L.H. 8" DISK DRIVE
R.H. 8" DISK DRIVE
J1
P1
P1
J1
P1A
J1
INDEX
DRIVE SEL 1
DRIVE SEL 2
READY
HEAD LOAD
DIR SEL
STEP
WRITE DATA
WRITE GATE
TRACK 00
WRITE PROTECT
READ DATA
SID SEL
INDEX
DRIVE SEL 1
DRIVE SEL 2
READY
HEAD LOAD
DIR SEL
STEP
WRITE DATA
WRITE GATE
TRACK 00
WRITE PROTECT
READ DATA
SID SEL


MAIN PWA
J1
P1
L.H. 5.25"
DISK DRIVE
P1
J1
R.H. 5.25"
DISK DRIVE
P1A
J1
INDEX
SEL-1
SEL-2
SID SEL
DIR SEL
STEP
WRITE DATA
WRITE GATE
TRACK 00
WRITE PROTECT
READ DATA
+5 VDC
+5V RET
+12 VDC
+12 RET
8/N5
+5VDC
+12VDC
+12RET
8
10
12
32
18
20
22
24
26
28
30

Notes

Theory of Operation

The display processor houses the system board, disk drive daughter board, the CRT, the power supply, and one bus expansion slot.

The system board has the following:

- Central Processing Unit (CPU)
- 6 to 8k of Read Only Memory (ROM)
- 64k of Random Access (Read/Write) Memory (RAM)
- Counter Timer Circuit (CTC)
- Serial Input/Output Controller (SIO)
- Parallel Input/Output Controller (PIO)
- Two RS-232-C Serial I/O Ports
- Dual 8-bit Parallel Ports
- CRT Controller and CRT Refresh Memory
- Speaker
- Disk Drive Daughter Board Connector
- Bus Expansion Connector
    - 8086 Co-processor (16/8 system)
- Parallel Keyboard Interface

CPU

The CPU is a Zilog Z80-A operating with a clock rate of 4 Mhz. It is initialized to use Interrupt Mode 2 by the ROSR monitor at power on. The Z80-A also provides refresh to the 64k of dynamic memory on the system board. Therefore, the I and R registers should not be altered by an application program.

ROM and RAM Memory

The System Board has two banks of memory. Bank 1 has 64k of RAM. Bank 0 has up to 8K of ROM.

When power is turned on or RESET is pressed, the Monitor, ROM/CRT RAM (Bank 0), is enabled by the hardware and the contents of the monitor ROM are moved by the CPU to the program memory starting at location F000H. When the move is complete, the CPU transfers control to location F000H and RAM (Bank 1) is enabled. Bank 0 is also enabled when a character is sent to the screen.

6-8k ROM

The CPU board has provisions for 4-2k x 8 Read Only Memory devices. The first 3 (U33, U34 & U35) store the firmware for the ROSR monitor. The fourth (U36) provides translation tables and related firmware for the position-encoded low profile keyboard.

64k RAM

The 64k byte (65536 x 8) RAM provides space for a portion of the ROSR monitor (upper 4k F000h - FFFFh), and 60k (0000h - EFFFh) is free for programs to execute in such as an operating system and an application program. This RAM is dynamic and refresh is provided by the Z80-A CPU.

Counter Timer Circuit (CTC)

The CTC has four independently-programmable counter/timer channels, each with a readable downcounter and a selectable 16 or 256 prescaler. Downcounters are reloaded automatically at zero. Each channel is programmed with two bytes. Once started, the CTC counts down, reloads its time constant automatically, and resumes counting. Internally, the CTC generates a unique vector for each channel.

Serial Input/Output Controller (SIO)

The Serial I/O Controller has two independent, full-duplex channels with separate control and status lines for modems or other devices. Data rates are from 50 to 19,200 bits/second. Channel A (modem) supports both Asynchronous and Synchronous protocols. Channel B (printer) is dedicated to Asynchronous. The receiver is quadruple-buffered and the transmitter is double-buffered. The controller also supports daisy-chain interrupt vectoring for interrupts without external logic.

Serial I/O Ports

The 820-II CPU board contains a Z80-A SIO that provides two user-accessible serial ports to the 25-pin printer and modem connectors on the rear of the display processor. The Communications port is capable of operating in synchronous or asynschronous modes, while the Printer port is only capable of operating asynchronously. On an Etch 2 CPU, there is a 30-pin connector. Selection of synchronous or asynchronous mode is under program control as opposed to the Etch 1 CPU (with a 40-pin connector) where a physical change is required to make the sync or async selection.

Parallel Input/Output Controller (PIO)

There is a System and a General Purpose Parallel I/O Controller which provides direct interface between the CPU and the peripheral devices. Each controller has two 8-bit I/O ports. The System PIO is dedicated for keyboard input, memory bank and CRT font selection, and floppy disk drive and side selection. The General Purpose PIO provides the user with a dual 8-bit parallel I/O port for interfacing with peripherals.

Parallel Port

The Z80-A General Purpose PIO is accessible on the main CPU board on connector J8. This PIO is programmed by the ROSR monitor at power-on to provide a parallel Centronics-compatible interface for a parallel printer. A transceiver is physically located between the Z80-A PIO and the J8 connector. Jumpers must be installed on option connector J11 to select whether the transceiver will transmit or receive data. See also page 24.

CRT Controller

The CPU board contains the 2k of refresh RAM where the characters that are to be displayed on the screen are stored. It also has the necessary electronics to provide the control signals (sync and video) to the CRT monitor. The CPU board has two character font ROMs; each font ROM contains two character sets.

U57	Normal white on black font
	Reverse video font
U58	Normal white on black font
	Graphic character font

The CRT driver in the ROSR monitor translates character-level escape sequences into commands as to which of the font ROMs to select and which of the two fonts inside the selected font ROM to select. Basically, characters that are stored in the CRT's refresh memory address the selected font ROM; the font ROM provides dot information to the video input of the CRT so the character can be displayed.

The characters on the CRT can have **one** of the following attributes:

Blink
Inverse video
Graphics
Low intensity

The most significant bit of the character stored in the CRT's refresh memory determines if the character is to be displayed with its attribute enabled.

The ROSR monitor provides a character-oriented command format for controlling the screen and font ROM selection. It is recommended that programs use this method to control the CRT and its attributes.

CRT RAM

Memory Allocation

The CRT RAM occupies 3000H - 3FFFH in bank 0 (System Bank). Each 80-character line on the CRT is allocated 128 bytes in the CRT RAM. Listed below are the starting and ending addresses for each of the 24 rows in the CRT RAM. The example (at the bottom) shows some character locations in CRT memory. (Assumes scroll register = 23)

Row	Starting Address	Ending Address
0	3000H	304FH
1	3080H	30CFH
2	3100H	314FH
3	3180H	31CFH
4	3200H	324FH
5	3280H	32CFH
6	3300H	334FH
7	3380H	33CFH
8	3400H	344FH
9	3480H	34CFH
10	3500H	354FH
11	3580H	35CFH
12	3600H	364FH
13	3680H	36CFH
14	3700H	374FH
15	3780H	37CFH
16	3800H	384FH
17	3880H	38CFH
18	3900H	394FH
19	3980H	39CFH
20	3A00H	3A4FH
21	3A80H	3ACFH
22	3B00H	3B4FH
23	3B80H	3BCFH

Row	Column	CRT Memory Address
0	0	3000H
0	79	304FH
1	1	3081H
1	5	3085H
23	0	3B80H
23	1	3B81H
23	79	3BCFH

Scroll Register

To eliminate the delay associated with software scrolling, hardware scrolling is employed. Writing into the scroll register (Port 14h) adds an offset to the line address developed by the line counter. For instance, with an offset of zero (scroll register = 0), the data at location 3000H (in the CRT refresh memory) will be displayed on the bottom row of the display. If the offset is 23, the data at location 3000H will be displayed on the top row of the screen. The scroll register is loaded from A8 to A15 rather than D0 to D7. Therefore, the scroll value must be in the B register if an indirect OUT instruction is used.

Scroll Register	Row 0, Column 0	Row 23, Column 0
23	3000H	3B80H
22	3080H	3B00H
21	3100H	3A80H
20	3180H	3A00H
19	3200H	3980H
18	3280H	3900H
17	3300H	3880H
16	3380H	3800H
15	3400H	3780H
14	3480H	3700H
13	3500H	3680H
12	3580H	3600H
11	3600H	3580H
10	3680H	3500H
9	3700H	3480H
8	3780H	3400H
7	3800H	3380H
6	3880H	3300H
5	3900H	3280H
4	3980H	3200H
3	3A00H	3180H
2	3A80H	3100H
1	3B00H	3080H
0	3B80H	3000H

Speaker
The 820-II and 16/8 have an audio speaker connected to two I/O ports (28h and 29h). Outputting to one I/O port causes the speaker cone to be pushed out; outputting to the other I/O port pulls in the speaker cone. The actual value output to these ports has no significance. To generate a beep, the application program can simply send an ASCII Bell character to the CRT. To generate a tone other than the standard bell character, the program must move the speaker cone in and out at the desired frequency.

Disk Drive Daughter Board
The disk drive connector on the rear is a "dual personality" connector, depending on which disk drive daughter board is installed on the mother board. Presently, there are two types of disk interface:

Shugart SASI interface controller suitable for interfacing to a SA1403D Rigid Disk Controller.

Floppy-only interface suitable for interfacing to Shugart SA800/SA400L/SA850/SA450 dual daisy-chained disk configurations.

The ROSR monitor detects which daughter board is installed at power-on and selects the appropriate physical disk driver firmware to process physical disk drive requests.

Caution:
If a rigid disk drive unit (U07, U08) is connected to a floppy display/processor (U03/H69, U04), the rigid controller PWB will be destroyed when power is switched on. The rigid disk drive unit must be connected only to a rigid display/processor (U05/H70, U06). Connecting a floppy disk drive unit (929/T66/973/F10, E41/E44/E42/E89) to a rigid display/processor (U05/H70, U06) may cause the processor PWB to fail. Before connecting any disk drive unit to a display/processor, check that the configuration of the display/processor is compatible with the disk drive unit. The configuration can be determined in one of two ways. (1) Check the product code of the display/processor. The product code is the first three digits of the serial number, located on the underside of the display processor. (2) Verify that the proper drive interface PWB is installed by checking the part number.

System Bus
The System Bus contains an 8-bit Data Bus (Tri-state, Input/Output) bi-directional Data exchange with memory and I/O devices. It has a 16-bit Address Bus to address up to 64k of memory for I/O devices data exchange.

Keyboard Interface
The keyboard FIFO (Etch 2 CPU only) has space for 16 (decimal) entries. Associated with the keyboard FIFO are input and output position pointers and a count of the number of entries currently in the FIFO.

The available memory pointers provide the addresses bounding the available unused RAM in the memory reserved for system use. Although these pointers are a supported feature, there is no guaranteed available memory size.

There are two tables used to disk map a logical disk to its physical driver. The first table, Seltab, associates a logical disk number with a physical disk number. The second table, Drvtab, identifies which physical disk driver is appropriate to use with the selected physical disk.

The physical driver command block is a collection of all information necessary for the disk system to perform the requested disk activity.

The timer and clock variables are a collection of locations used for maintaining the one second timer and the time-of-day clock and calendar. The console command line buffer immediately follows these variables.

Z8400
Z80® CPU Central Processing Unit

Zilog

Product Specification

June 1982

Features

- The instruction set contains 158 instructions. The 78 instructions of the 8080A are included as a subset; 8080A software compatibility is maintained.
- Six MHz, 4 MHz and 2.5 MHz clocks for the Z80B, Z80A, and Z80 CPU result in rapid instruction execution with consequent high data throughput.
- The extensive instruction set includes string, bit, byte, and word operations. Block searches and block transfers together with indexed and relative addressing result in the most powerful data handling capabilities in the microcomputer industry.
- The Z80 microprocessors and associated family of peripheral controllers are linked by a vectored interrupt system. This system may be daisy-chained to allow implementation of a priority interrupt scheme. Little, if any, additional logic is required for daisy-chaining.
- Duplicate sets of both general-purpose and flag registers are provided, easing the design and operation of system software through single-context switching, background-foreground programming, and single-level interrupt processing. In addition, two 16-bit index registers facilitate program processing of tables and arrays.
- There are three modes of high speed interrupt processing: 8080 compatible, non-Z80 peripheral device, and Z80 Family peripheral with or without daisy chain.
- On-chip dynamic memory refresh counter.

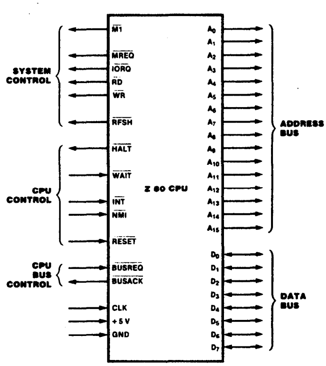


Figure 1. Pin Functions

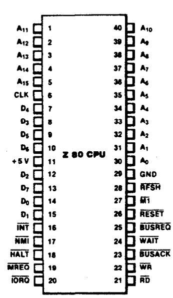


Figure 2. Pin Assignments

General Description

The Z80, Z80A, and Z80B CPUs are third-generation single-chip microprocessors with exceptional computational power. They offer higher system throughput and more efficient memory utilization than comparable second- and third-generation microprocessors. The internal registers contain 208 bits of read/write memory that are accessible to the programmer. These registers include two sets of six general-purpose registers which may be used individually as either 8-bit registers or as 16-bit register pairs. In addition, there are two sets of accumulator and flag registers. A group of "Exchange" instructions makes either set of main or alternate registers accessible to the programmer. The alternate set allows operation in foreground-background mode or it may be reserved for very fast interrupt response.

The Z80 also contains a Stack Pointer, Program Counter, two index registers, a Refresh register (counter), and an Interrupt register. The CPU is easy to incorporate into a system since it requires only a single +5 V power source. All output signals are fully decoded and timed to control standard memory or peripheral circuits, and it is supported by an extensive family of peripheral controllers. The internal block diagram (Figure 3) shows the primary functions of the Z80 processors. Subsequent text provides more detail on the Z80 I/O controller family, registers, instruction set, interrupts and daisy chaining, and CPU timing.

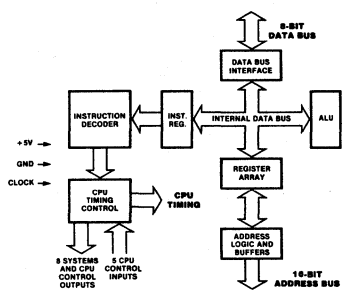


Figure 3. Z80 CPU Block Diagram

Z80 Microprocessor Family

The Zilog Z80 microprocessor is the central element of a comprehensive microprocessor product family. This family works together in most applications with minimum requirements for additional logic, facilitating the design of efficient and cost-effective microcomputer-based systems.

Zilog has designed five components to provide extensive support for the Z80 microprocessor. These are:

- The PIO (Parallel Input/Output) operates in both data-byte I/O transfer mode (with handshaking) and in bit mode (without handshaking). The PIO may be configured to interface with standard parallel peripheral devices such as printers, tape punches, and keyboards.
- The CTC (Counter/Timer Circuit) features four programmable 8-bit counter/timers, each of which has an 8-bit prescaler. Each of the four channels may be configured to operate in either counter or timer mode.
- The DMA (Direct Memory Access) controller provides dual port data transfer operations and the ability to terminate data transfer as a result of a pattern match.
- The SIO (Serial Input/Output) controller offers two channels. It is capable of operating in a variety of programmable modes for both synchronous and asynchronous communication, including Bi-Sync and SDLC.
- The DART (Dual Asynchronous Receiver/Transmitter) device provides low cost asynchronous serial communication. It has two channels and a full modem control interface.

Z80 CPU Registers

Figure 4 shows three groups of registers within the Z80 CPU. The first group consists of duplicate sets of 8-bit registers: a principal set and an alternate set (designated by ' [prime], e.g., A'). Both sets consist of the Accumulator Register, the Flag Register, and six general-purpose registers. Transfer of data between these duplicate sets of registers is accomplished by use of "Exchange" instructions. The result is faster response to interrupts and easy, efficient implementation of such versatile programming techniques as background-foreground data processing. The second set of registers consists of six registers with assigned functions. These are the I (Interrupt Register), the R (Refresh Register), the IX and IY (Index Registers), the SP (Stack Pointer), and the PC (Program Counter). The third group consists of two interrupt status flip-flops, plus an additional pair of flip-flops which assists in identifying the interrupt mode at any particular time. Table 1 provides further information on these registers.

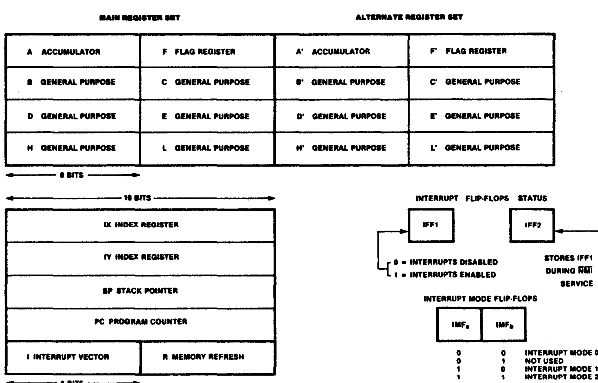


Figure 4. CPU Registers

Z80 CPU Registers (Continued)

Register		Size (Bits)	Remarks
A, A'	Accumulator	8	Stores an operand or the results of an operation.
F, F'	Flags	8	See Instruction Set.
B, B'	General Purpose	8	Can be used separately or as a 16-bit register with C.
C, C'	General Purpose	8	See B, above.
D, D'	General Purpose	8	Can be used separately or as a 16-bit register with E.
E, E'	General Purpose	8	See D, above.
H, H'	General Purpose	8	Can be used separately or as a 16-bit register with L.
L, L'	General Purpose	8	See H, above.
			Note: The (B,C), (D,E), and (H,L) sets are combined as follows: B — High byte C — Low byte D — High byte E — Low byte H — High byte L — Low byte
I	Interrupt Register	8	Stores upper eight bits of memory address for vectored interrupt processing.
R	Refresh Register	8	Provides user-transparent dynamic memory refresh. Automatically incremented and placed on the address bus during each instruction fetch cycle.
IX	Index Register	16	Used for indexed addressing.
IY	Index Register	16	Same as IX, above.
SP	Stack Pointer	16	Holds address of the top of the stack. See Push or Pop in instruction set.
PC	Program Counter	16	Holds address of next instruction.
IFF_1-IFF_2	Interrupt Enable	Flip-Flops	Set or reset to indicate interrupt status (see Figure 4).
IMFa-IMFb	Interrupt Mode	Flip-Flops	Reflect Interrupt mode (see Figure 4).

Table 1. Z80 CPU Registers

Interrupts: General Operation

The CPU accepts two interrupt input signals: $\overline{\text{NMI}}$ and $\overline{\text{INT}}$. The $\overline{\text{NMI}}$ is a non-maskable interrupt and has the highest priority. $\overline{\text{INT}}$ is a lower priority interrupt and it requires that interrupts be enabled in software in order to operate. $\overline{\text{INT}}$ can be connected to multiple peripheral devices in a wired-OR configuration.

The Z80 has a single response mode for interrupt service for the non-maskable interrupt. The maskable interrupt, $\overline{\text{INT}}$, has three programmable response modes available. These are:

- Mode 0 — compatible with the 8080 microprocessor.
- Mode 1 — Peripheral Interrupt service, for use with non-8080/Z80 systems.
- Mode 2 — a vectored interrupt scheme, usually daisy-chained, for use with Z80 Family and compatible peripheral devices.

The CPU services interrupts by sampling the $\overline{\text{NMI}}$ and $\overline{\text{INT}}$ signals at the rising edge of the last clock of an instruction. Further interrupt service processing depends upon the type of interrupt that was detected. Details on interrupt responses are shown in the CPU Timing Section.

Interrupts: General Operation (Continued)

Non-Maskable Interrupt ($\overline{\text{NMI}}$). The non-maskable interrupt cannot be disabled by program control and therefore will be accepted at all times by the CPU. $\overline{\text{NMI}}$ is usually reserved for servicing only the highest priority type interrupts, such as that for orderly shut-down after power failure has been detected. After recognition of the $\overline{\text{NMI}}$ signal (providing $\overline{\text{BUSREQ}}$ is not active), the CPU jumps to restart location 0066H. Normally, software starting at this address contains the interrupt service routing.

Maskable Interrupt ($\overline{\text{INT}}$). Regardless of the interrupt mode set by the user, the Z80 response to a maskable interrupt input follows a common timing cycle. After the interrupt has been detected by the CPU (provided that interrupts are enabled and $\overline{\text{BUSREQ}}$ is not active) a special interrupt processing cycle begins. This is a special fetch ($\overline{\text{M1}}$) cycle in which $\overline{\text{IORQ}}$ becomes active rather than $\overline{\text{MREQ}}$, as in normal $\overline{\text{M1}}$ cycle. In addition, this special $\overline{\text{M1}}$ cycle is automatically extended by two $\overline{\text{WAIT}}$ states, to allow for the time required to acknowledge the interrupt request.

Mode 0 Interrupt Operation. This mode is compatible with the 8080 microprocessor interrupt service procedures. The interrupting device places an instruction on the data bus. This is normally a Restart Instruction, which will initiate a call to the selected one of eight restart locations in page zero of memory.

Mode 1 Interrupt Operation. Mode 1 operation is very similar to that for the $\overline{\text{NMI}}$. The principal difference is that the Mode 1 interrupt has a restart location of 0038H only.

Mode 2 Interrupt Operation. This interrupt mode has been designed to utilize most effectively the capabilities of the Z80 microprocessor and its associated peripheral family. The interrupting peripheral device selects the starting address of the interrupt service routine. It does this by placing an 8-bit vector on the data bus during the interrupt acknowledge cycle. The CPU forms a pointer using this byte as the lower 8-bits and the contents of the I register as the upper 8-bits. This points to an entry in a table of addresses for interrupt service routines. The CPU then jumps to the routine at that address. This flexibility in selecting the interrupt service routine address allows the peripheral device to use several different types of service routines. These routines may be located at any available location in memory. Since the interrupting device supplies the low-order byte of the 2-byte vector, bit 0 (A_0) must be a zero.

Interrupt Priority (Daisy Chaining and Nested Interrupts). The interrupt priority of each peripheral device is determined by its physical location within a daisy-chain configuration. Each device in the chain has an interrupt enable input line (IEI) and an interrupt enable output line (IEO), which is fed to the next lower priority device. The first device in the daisy chain has its IEI input hardwired to a High level. The first device has highest priority, while each succeeding device has a corresponding lower priority. This arrangement permits the CPU to select the highest priority interrupt from several simultaneously interrupting peripherals.

The interrupting device disables its IEO line to the next lower priority peripheral until it has been serviced. After servicing, its IEO line is raised, allowing lower priority peripherals to demand interrupt servicing.

The Z80 CPU will nest (queue) any pending interrupts or interrupts received while a selected peripheral is being serviced.

Interrupt Enable/Disable Operation. Two flip-flops, IFF_1 and IFF_2, referred to in the register description are used to signal the CPU interrupt status. Operation of the two flip-flops is described in Table 2. For more details, refer to the *Z80 CPU Technical Manual* and *Z80 Assembly Language Manual.*

Action	IFF_1	IFF_2	Comments
CPU Reset	0	0	Maskable interrupt $\overline{\text{INT}}$ disabled
DI instruction execution	0	0	Maskable interrupt $\overline{\text{INT}}$ disabled
EI instruction execution	1	1	Maskable interrupt $\overline{\text{INT}}$ enabled
LD A,I instruction execution	•	•	$IFF_2 \rightarrow$ Parity flag
LD A,R instruction execution	•	•	$IFF_2 \rightarrow$ Parity flag
Accept $\overline{\text{NMI}}$	0	IFF_1	$IFF_1 \rightarrow IFF_2$ (Maskable interrupt $\overline{\text{INT}}$ disabled)
RETN instruction execution	IFF_2	•	$IFF_2 \rightarrow IFF_1$ at completion of an $\overline{\text{NMI}}$ service routine.

Table 2. State of Flip-Flops

Instruction Set

The Z80 microprocessor has one of the most powerful and versatile instruction sets available in any 8-bit microprocessor. It includes such unique operations as a block move for fast, efficient data transfers within memory or between memory and I/O. It also allows operations on any bit in any location in memory.

The following is a summary of the Z80 instruction set and shows the assembly language mnemonic, the operation, the flag status, and gives comments on each instruction. The *Z80 CPU Technical Manual* (03-0029-01) and *Assembly Language Programming Manual* (03-0002-01) contain significantly more details for programming use.

The instructions are divided into the following categories:

- □ 8-bit loads
- □ 16-bit loads
- □ Exchanges, block transfers, and searches
- □ 8-bit arithmetic and logic operations
- □ General-purpose arithmetic and CPU control
- □ 16-bit arithmetic operations
- □ Rotates and shifts
- □ Bit set, reset, and test operations
- □ Jumps
- □ Calls, returns, and restarts
- □ Input and output operations

A variety of addressing modes are implemented to permit efficient and fast data transfer between various registers, memory locations, and input/output devices. These addressing modes include:

- □ Immediate
- □ Immediate extended
- □ Modified page zero
- □ Relative
- □ Extended
- □ Indexed
- □ Register
- □ Register indirect
- □ Implied
- □ Bit

8-Bit Load Group

Mnemonic	Symbolic Operation	S	Z		Flags H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
LD r, r'	r ← r'	•	•	X	•	X	•	•	•	01 r r'		1	1	4	r, r' Reg.
LD r, n	r ← n	•	•	X	•	X	•	•	•	00 r 110 ← n →		2	2	7	000 B 001 C
LD r, (HL)	r ← (HL)	•	•	X	•	X	•	•	•	01 r 110		1	2	7	010 D
LD r, (IX+d)	r ← (IX+d)	•	•	X	•	X	•	•	•	11 011 101 01 r 101 ← d →	DD	3	5	19	011 E 100 H 101 L
LD r, (IY+d)	r ← (IY+d)	•	•	X	•	X	•	•	•	11 111 101 01 r 110 ← d →	FD	3	5	19	111 A
LD (HL), r	(HL) ← r	•	•	X	•	X	•	•	•	01 110 r		1	2	7	
LD (IX+d), r	(IX+d) ← r	•	•	X	•	X	•	•	•	11 011 101 01 110 r ← d →	DD	3	5	19	
LD (IY+d), r	(IY+d) ← r	•	•	X	•	X	•	•	•	11 111 101 01 110 r ← d →	FD	3	5	19	
LD (HL), n	(HL) ← n	•	•	X	•	X	•	•	•	00 110 110 ← n →	36	2	3	10	
LD (IX+d), n	(IX+d) ← n	•	•	X	•	X	•	•	•	11 011 101 00 110 110 ← d → ← n →	DD 36	4	5	19	
LD (IY+d), n	(IY+d) ← n	•	•	X	•	X	•	•	•	11 111 101 00 110 110 ← d → ← n →	FD 36	4	5	19	
LD A, (BC)	A ← (BC)	•	•	X	•	X	•	•	•	00 001 010	0A	1	2	7	
LD A, (DE)	A ← (DE)	•	•	X	•	X	•	•	•	00 011 010	1A	1	2	7	
LD A, (nn)	A ← (nn)	•	•	X	•	X	•	•	•	00 111 010 ← n → ← n →	3A	3	4	13	
LD (BC), A	(BC) ← A	•	•	X	•	X	•	•	•	00 000 010	02	1	2	7	
LD (DE), A	(DE) ← A	•	•	X	•	X	•	•	•	00 010 010	12	1	2	7	
LD (nn), A	(nn) ← A	•	•	X	•	X	•	•	•	00 110 010 ← n → ← n →	32	3	4	13	
LD A, I	A ← I	↕	↕	X	0	X	IFF	0	•	11 101 101 01 010 111	ED 57	2	2	9	
LD A, R	A ← R	↕	↕	X	0	X	IFF	0	•	11 101 101 01 011 111	ED 5F	2	2	9	
LD I, A	I ← A	•	•	X	•	X	•	•	•	11 101 101 01 000 111	ED 47	2	2	9	
LD R, A	R ← A	•	•	X	•	X	•	•	•	11 101 101 01 001 111	ED 4F	2	2	9	

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.
For an explanation of flag notation and symbols for mnemonic tables, see Symbolic Notation section following tables.

16-Bit Load Group

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
LD dd, nn	dd ← nn	•	•	X	•	X	•	•	•	00 dd0 001 ← n → ← n →		3	3	10	dd Pair 00 BC 01 DE 10 HL 11 SP
LD IX, nn	IX ← nn	•	•	X	•	X	•	•	•	11 011 101 00 100 001 ← n → ← n →	DD 21	4	4	14	
LD IY, nn	IY ← nn	•	•	X	•	X	•	•	•	11 111 101 00 100 001 ← n → ← n →	FD 21	4	4	14	
LD HL, (nn)	H ← (nn+1) L ← (nn)	•	•	X	•	X	•	•	•	00 101 010 ← n → ← n →	2A	3	5	16	
LD dd, (nn)	dd_H ← (nn+1) dd_L ← (nn)	•	•	X	•	X	•	•	•	11 101 101 01 dd1 011 ← n → ← n →	ED	4	6	20	
LD IX, (nn)	IX_H ← (nn+1) IX_L ← (nn)	•	•	X	•	X	•	•	•	11 011 101 00 101 010 ← n → ← n →	DD 2A	4	6	20	
LD IY, (nn)	IY_H ← (nn+1) IY_L ← (nn)	•	•	X	•	X	•	•	•	11 111 101 00 101 010 ← n → ← n →	FD 2A	4	6	20	
LD (nn), HL	(nn+1) ← H (nn) ← L	•	•	X	•	X	•	•	•	00 100 010 ← n → ← n →	22	3	5	16	
LD (nn), dd	(nn+1) ← dd_H (nn) ← dd_L	•	•	X	•	X	•	•	•	11 101 101 01 dd0 011 ← n → ← n →	ED	4	6	20	
LD (nn), IX	(nn+1) ← IX_H (nn) ← IX_L	•	•	X	•	X	•	•	•	11 011 101 00 100 010 ← n → ← n →	DD 22	4	6	20	
LD (nn), IY	(nn+1) ← IY_H (nn) ← IY_L	•	•	X	•	X	•	•	•	11 111 101 00 100 010 ← n → ← n →	FD 22	4	6	20	
LD SP, HL	SP ← HL	•	•	X	•	X	•	•	•	11 111 001	F9	1	1	6	
LD SP, IX	SP ← IX	•	•	X	•	X	•	•	•	11 011 101 11 111 001	DD F9	2	2	10	
LD SP, IY	SP ← IY	•	•	X	•	X	•	•	•	11 111 101 11 111 001	FD F9	2	2	10	
PUSH qq	(SP−2) ← qq_L (SP−1) ← qq_H SP ← SP −2	•	•	X	•	X	•	•	•	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 SP ← SP −2	•	•	X	•	X	•	•	•	11 011 101 11 100 101	DD E5	2	4	15	
PUSH IY	(SP−2) ← IY_L (SP−1) ← IY_H SP ← SP −2	•	•	X	•	X	•	•	•	11 111 101 11 100 101	FD E5	2	4	15	
POP qq	qq_H ← (SP+1) qq_L ← (SP) SP ← SP +2	•	•	X	•	X	•	•	•	11 qq0 001		1	3	10	
POP IX	IX_H ← (SP+1) IX_L ← (SP) SP ← SP +2	•	•	X	•	X	•	•	•	11 011 101 11 100 001	DD E1	2	4	14	
POP IY	IY_H ← (SP+1) IY_L ← (SP) SP ← SP +2	•	•	X	•	X	•	•	•	11 111 101 11 100 001	FD E1	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.

Exchange, Block Transfer, Block Search Groups

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
EX DE, HL	DE ↔ HL	•	•	X	•	X	•	•	•	11 101 011	EB	1	1	4	
EX AF, AF'	AF ↔ AF'	•	•	X	•	X	•	•	•	00 001 000	08	1	1	4	
EXX	BC ↔ BC' DE ↔ DE' HL ↔ HL'	•	•	X	•	X	•	•	•	11 011 001	D9	1	1	4	Register bank and auxiliary register bank exchange
EX (SP), HL	H ↔ (SP+1) L ↔ (SP)	•	•	X	•	X	•	•	•	11 100 011	E3	1	5	19	
EX (SP), IX	IX_H ↔ (SP+1) IX_L ↔ (SP)	•	•	X	•	X	•	•	•	11 011 101 11 100 011	DD E3	2	6	23	
EX (SP), IY	IY_H ↔ (SP+1) IY_L ↔ (SP)	•	•	X	•	X	•	•	•	11 111 101 11 100 011	FD E3	2	6	23	
LDI	(DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC−1	•	•	X	0	X	① ‡	0	•	11 101 101 10 100 000	ED A0	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	•	•	X	0	X	0	0	•	11 101 101 10 110 000	ED B0	2 2	5 4	21 16	If BC ≠ 0 If BC = 0

NOTE: ① P/V flag is 0 if the result of BC − 1 = 0, otherwise P/V = 1.

Exchange, Block Transfer, Block Search Groups (Continued)

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
LDD	(DE) ← (HL) DE ← DE − 1 HL ← HL − 1 BC ← BC − 1	•	•	X	0	X	① ↕	0	•	11 101 101 10 101 000	ED A8	2	4	16	
LDDR	(DE) ← (HL) DE ← DE − 1 HL ← HL − 1 BC ← BC − 1 Repeat until BC = 0	•	•	X	0	X	0	0	•	11 101 101 10 111 000	ED B8	2 2	5 4	21 16	If BC ≠ 0 If BC = 0
CPI	A − (HL) HL ← HL + 1 BC ← BC − 1	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 100 001	ED A1	2	4	16	
CPIR	A − (HL) HL ← HL + 1 BC ← BC − 1 Repeat until A = (HL) or BC = 0	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 110 001	ED B1	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 ↕	X	↕	X	① ↕	1	•	11 101 101 10 101 001	ED A9	2	4	16	
CPDR	A − (HL) HL ← HL − 1 BC ← BC − 1 Repeat until A = (HL) or BC = 0	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 111 001	ED B9	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 if A = (HL), otherwise Z = 0.

8-Bit Arithmetic and Logical Group

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
ADD A, r	A ← A + r	↕	↕	X	↕	X	V	0	↕	10 [000] r		1	1	4	r Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
ADD A, n	A ← A + n	↕	↕	X	↕	X	V	0	↕	11 [000] 110 ← n →		2	2	7	
ADD A, (HL)	A ← A + (HL)	↕	↕	X	↕	X	V	0	↕	10 [000] 110		1	2	7	
ADD A, (IX+d)	A ← A + (IX+d)	↕	↕	X	↕	X	V	0	↕	11 011 101 10 [000] 110 ← d →	DD	3	5	19	
ADD A, (IY+d)	A ← A + (IY+d)	↕	↕	X	↕	X	V	0	↕	11 111 101 10 [000] 110 ← d →	FD	3	5	19	
ADC A, s	A ← A + s + CY	↕	↕	X	↕	X	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	↕	↕	X	↕	X	V	1	↕	[010]					
SBC A, s	A ← A − s − CY	↕	↕	X	↕	X	V	1	↕	[011]					
AND s	A ← A ∧ s	↕	↕	X	1	X	P	0	0	[100]					
OR s	A ← A ∨ s	↕	↕	X	0	X	P	0	0	[110]					
XOR s	A ← A ⊕ s	↕	↕	X	0	X	P	0	0	[101]					
CP s	A − s	↕	↕	X	↕	X	V	1	↕	[111]					
INC r	r ← r + 1	↕	↕	X	↕	X	V	0	•	00 r [100]		1	1	4	
INC (HL)	(HL) ← (HL) + 1	↕	↕	X	↕	X	V	0	•	00 110 [100]		1	3	11	
INC (IX+d)	(IX+d) ← (IX+d) + 1	↕	↕	X	↕	X	V	0	•	11 011 101 00 110 [100] ← d →	DD	3	6	23	
INC (IY+d)	(IY+d) ← (IY+d) + 1	↕	↕	X	↕	X	V	0	•	11 111 101 00 110 [100] ← d →	FD	3	6	23	
DEC m	m ← m − 1	↕	↕	X	↕	X	V	1	•	[101]					m is any of r, (HL), (IX+d), (IY+d) as shown for INC. DEC same format and states as INC. Replace [100] with [101] in opcode.

General-Purpose Arithmetic and CPU Control Groups

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
DAA	Converts acc. content into packed BCD following add or subtract with packed BCD operands.	↕	↕	X	↕	X	P	•	↕	00 100 111	27	1	1	4	Decimal adjust accumulator.
CPL	A ← $\overline{A}$	•	•	X	1	X	•	1	•	00 101 111	2F	1	1	4	Complement accumulator (one's complement).
NEG	A ← 0 − A	↕	↕	X	↕	X	V	1	↕	11 101 101 01 000 100	ED 44	2	2	8	Negate acc. (two's complement).
CCF	CY ← $\overline{CY}$	•	•	X	X	X	•	0	↕	00 111 111	3F	1	1	4	Complement carry flag.
SCF	CY ← 1	•	•	X	0	X	•	0	1	00 110 111	37	1	1	4	Set carry flag.
NOP	No operation	•	•	X	•	X	•	•	•	00 000 000	00	1	1	4	
HALT	CPU halted	•	•	X	•	X	•	•	•	01 110 110	76	1	1	4	
DI ★	IFF ← 0	•	•	X	•	X	•	•	•	11 110 011	F3	1	1	4	
EI ★	IFF ← 1	•	•	X	•	X	•	•	•	11 111 011	FB	1	1	4	
IM 0	Set interrupt mode 0	•	•	X	•	X	•	•	•	11 101 101 01 000 110	ED 46	2	2	8	
IM 1	Set interrupt mode 1	•	•	X	•	X	•	•	•	11 101 101 01 010 110	ED 56	2	2	8	
IM 2	Set interrupt mode 2	•	•	X	•	X	•	•	•	11 101 101 01 011 110	ED 5E	2	2	8	

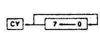
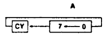
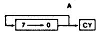
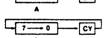
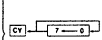
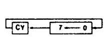
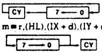
NOTES: IFF indicates the interrupt enable flip-flop.
CY indicates the carry flip-flop.
★ indicates interrupts are not sampled at the end of EI or DI.

16-Bit Arithmetic Group

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
ADD HL, ss	HL ← HL + ss	•	•	X	X	X	•	0	↕	00 ss1 001		1	3	11	ss Reg. 00 BC 01 DE 10 HL 11 SP
ADC HL, ss	HL ← HL + ss + CY	↕	↕	X	X	X	V	0	↕	11 101 101 01 ss1 010	ED	2	4	15	
SBC HL, ss	HL ← HL − ss − CY	↕	↕	X	X	X	V	1	↕	11 101 101 01 ss0 010	ED	2	4	15	
ADD IX, pp	IX ← IX + pp	•	•	X	X	X	•	0	↕	11 011 101 01 pp1 001	DD	2	4	15	pp Reg. 00 BC 01 DE 10 IX 11 SP
ADD IY, rr	IY ← IY + rr	•	•	X	X	X	•	0	↕	11 111 101 00 rr1 001	FD	2	4	15	rr Reg. 00 BC 01 DE 10 IY 11 SP
INC ss	ss ← ss + 1	•	•	X	•	X	•	•	•	00 ss0 011		1	1	6	
INC IX	IX ← IX + 1	•	•	X	•	X	•	•	•	11 011 101 00 100 011	DD 23	2	2	10	
INC IY	IY ← IY + 1	•	•	X	•	X	•	•	•	11 111 101 00 100 011	FD 23	2	2	10	
DEC ss	ss ← ss − 1	•	•	X	•	X	•	•	•	00 ss1 011		1	1	6	
DEC IX	IX ← IX − 1	•	•	X	•	X	•	•	•	11 011 101 00 101 011	DD 2B	2	2	10	
DEC IY	IY ← IY − 1	•	•	X	•	X	•	•	•	11 111 101 00 101 011	FD 2B	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.

Rotate and Shift Group

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
RLCA		•	•	X	0	X	•	0	↕	00 000 111	07	1	1	4	Rotate left circular accumulator.
RLA		•	•	X	0	X	•	0	↕	00 010 111	17	1	1	4	Rotate left accumulator.
RRCA		•	•	X	0	X	•	0	↕	00 001 111	0F	1	1	4	Rotate right circular accumulator.
RRA		•	•	X	0	X	•	0	↕	00 011 111	1F	1	1	4	Rotate right accumulator.
RLC r		↕	↕	X	0	X	P	0	↕	11 001 011 00 $\boxed{000}$ r	CB	2	2	8	Rotate left circular register r.
RLC (HL)		↕	↕	X	0	X	P	0	↕	11 001 011 00 $\boxed{000}$ 110	CB	2	4	15	r Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
RLC (IX + d)	 r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	11 011 101 11 001 011 ← d → 00 $\boxed{000}$ 110	DD CB	4	6	23	
RLC (IY + d)		↕	↕	X	0	X	P	0	↕	11 111 101 11 001 011 ← d → 00 $\boxed{000}$ 110	FD CB	4	6	23	
RL m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	$\boxed{010}$					Instruction format and states are as shown for RLC's. To form new opcode replace $\boxed{000}$ or RLC's with shown code.
RRC m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	$\boxed{001}$					

Rotate and Shift Group (Continued)

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
RR m	[7 → 0] → [CY] → (back to 7) m ≡ r, (HL), (IX + d), (IY + d)	↕	↕	X	0	X	P	0	↕	$\boxed{011}$					
SLA m	[CY] ← [7 ← 0] ← 0 m ≡ r, (HL), (IX + d), (IY + d)	↕	↕	X	0	X	P	0	↕	$\boxed{100}$					
SRA m	[7 → 0] → [CY] (bit 7 kept) m ≡ r, (HL), (IX + d), (IY + d)	↕	↕	X	0	X	P	0	↕	$\boxed{101}$					
SRL m	0 → [7 → 0] → [CY] m ≡ r, (HL), (IX + d), (IY + d)	↕	↕	X	0	X	P	0	↕	$\boxed{111}$					
RLD	A: [7-4 | 3-0] (HL): [7-4 | 3-0]	↕	↕	X	0	X	P	0	•	11 101 101 01 101 111	ED 6F	2	5	18	Rotate digit left and right between the accumulator and location (HL).
RRD	A: [7-4 | 3-0] → (HL): [7-4 | 3-0]	↕	↕	X	0	X	P	0	•	11 101 101 01 100 111	ED 67	2	5	18	The content of the upper half of the accumulator is unaffected.

Bit Set, Reset and Test Group

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
BIT b, r	$Z \leftarrow \overline{r}_b$	X	↕	X	1	X	X	0	•	11 001 011 01 b r	CB	2	2	8	r Reg. 000 B 001 C
BIT b, (HL)	$Z \leftarrow \overline{(HL)}_b$	X	↕	X	1	X	X	0	•	11 001 011 01 b 110	CB	2	3	12	010 D
BIT b, (IX+d)$_b$	$Z \leftarrow \overline{(IX+d)}_b$	X	↕	X	1	X	X	0	•	11 011 101 11 001 011 ← d → 01 b 110	DD CB	4	5	20	011 E 100 H 101 L 111 A b Bit Tested
BIT b, (IY+d)$_b$	$Z \leftarrow \overline{(IY+d)}_b$	X	↕	X	1	X	X	0	•	11 111 101 11 001 011 ← d → 01 b 110	FD CB	4	5	20	000 0 001 1 010 2 011 3 100 4 101 5 110 6 111 7
SET b, r	$r_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 001 011 $\boxed{11}$ b r	CB	2	2	8	
SET b, (HL)	$(HL)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 001 011 $\boxed{11}$ b 110	CB	2	4	15	
SET b, (IX+d)	$(IX+d)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 011 101 11 001 011 ← d → $\boxed{11}$ b 110	DD CB	4	6	23	
SET b, (IY+d)	$(IY+d)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 111 101 11 001 011 ← d → $\boxed{11}$ b 110	FD CB	4	6	23	
RES b, m	$m_b \leftarrow 0$ m ≡ r, (HL), (IX + d), (IY + d)	•	•	X	•	X	•	•	•	$\boxed{10}$					To form new opcode replace $\boxed{11}$ of SET b, s with $\boxed{10}$. Flags and time states for SET instruction.

NOTES: The notation m_b indicates bit b (0 to 7) or location m.

Jump Group

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
JP nn	PC ← nn	•	•	X	•	X	•	•	•	11 000 011 ← n → ← n →	C3	3	3	10	
JP cc, nn	If condition cc is true PC ← nn, otherwise continue	•	•	X	•	X	•	•	•	11 cc 010 ← n → ← n →		3	3	10	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
JR e	PC ← PC+e	•	•	X	•	X	•	•	•	00 011 000 ← e−2 →	18	2	3	12	111 M sign negative
JR C, e	If C = 0, continue	•	•	X	•	X	•	•	•	00 111 000 ← e−2 →	38	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	•	•	X	•	X	•	•	•	00 110 000 ← e−2 →	30	2	2	7	If condition not met.
	If C = 0, PC ← PC+e											2	3	12	If condition is met.
JP Z, e	If Z = 0 continue	•	•	X	•	X	•	•	•	00 101 000 ← e−2 →	28	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	•	•	X	•	X	•	•	•	00 100 000 ← e−2 →	20	2	2	7	If condition not met.
	If Z = 0, PC ← PC+e											2	3	12	If condition is met.
JP (HL)	PC ← HL	•	•	X	•	X	•	•	•	11 101 001	E9	1	1	4	
JP (IX)	PC ← IX	•	•	X	•	X	•	•	•	11 011 101 11 101 001	DD E9	2	2	8	

Jump Group (Continued)

Mnemonic	Symbolic Operation	S	Z		H		P/V	N ②	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
JP (IY)	PC ← IY	•	•	X	•	X	•	•	•	11 111 101 11 101 001	FD E9	2	2	8	
DJNZ, e	B ← B − 1 If B = 0, continue	•	•	X	•	X	•	•	•	00 010 000 ← e − 2 →	10	2	2	8	If B = 0.
	If B ≠ 0, PC ← PC + e											2	3	13	If B ≠ 0.

NOTES: e represents the extension in the relative addressing mode.
e is a signed two's complement number in the range < −126, 129 >.
e − 2 in the opcode provides an effective address of pc + e as PC is incremented by 2 prior to the addition of e.

Call and Return Group

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
CALL nn	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$ PC ← nn	•	•	X	•	X	•	•	•	11 001 101 ← n → ← n →	CD	3	5	17	
CALL cc, nn	If condition cc is false continue,	•	•	X	•	X	•	•	•	11 cc 100 ← n → ← n →		3	3	10	If cc is false.
	otherwise same as CALL nn											3	5	17	If cc is true.
RET	$PC_L \leftarrow (SP)$ $PC_H \leftarrow (SP+1)$	•	•	X	•	X	•	•	•	11 001 001	C9	1	3	10	
RET cc	If condition cc is false continue,	•	•	X	•	X	•	•	•	11 cc 000		1	1	5	If cc is false.
	otherwise same as RET											1	3	11	If cc is true. 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
RETI	Return from interrupt	•	•	X	•	X	•	•	•	11 101 101 01 001 101	ED 4D	2	4	14	
RETN[1]	Return from non-maskable interrupt	•	•	X	•	X	•	•	•	11 101 101 01 000 101	ED 45	2	4	14	
RST p	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$ $PC_H \leftarrow 0$ $PC_L \leftarrow p$	•	•	X	•	X	•	•	•	11 t 111		1	3	11	t p 000 00H 001 08H 010 10H 011 18H 100 20H 101 28H 110 30H 111 38H

NOTE: [1]RETN loads $IFF_2 \leftarrow IFF_1$

Input and Output Group

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
IN A, (n)	A ← (n)	•	•	X	•	X	•	•	•	11 011 011 ← n →	DB	2	3	11	n to $A_0 \sim A_7$ Acc. to $A_8 \sim A_{15}$
IN r, (C)	r ← (C) if r = 110 only the flags will be affected	↕	↕	X	↕	X	P	0	•	11 101 101 01 r 000	ED	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INI	(HL) ← (C) B ← B − 1 HL ← HL + 1	X	① ↕	X	X	X	X	↕	X	11 101 101 10 100 010	ED A2	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INIR	(HL) ← (C) B ← B − 1 HL ← HL + 1 Repeat until B = 0	X	1	X	X	X	X	↕	X	11 101 101 10 110 010	ED B2	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
IND	(HL) ← (C) B ← B − 1 HL ← HL − 1	X	① ↕	X	X	X	X	↕	X	11 101 101 10 101 010	ED AA	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INDR	(HL) ← (C) B ← B − 1 HL ← HL − 1 Repeat until B = 0	X	1	X	X	X	X	↕	X	11 101 101 10 111 010	ED BA	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUT (n), A	(n) ← A	•	•	X	•	X	•	↕	X	11 010 011 ← n →	D3	2	3	11	n to $A_0 \sim A_7$ Acc. to $A_8 \sim A_{15}$
OUT (C), r	(C) ← r	•	•	X	•	X	•	↕	X	11 101 101 01 r 001	ED	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTI	(C) ← (HL) B ← B − 1 HL ← HL + 1	X	① ↕	X	X	X	X	↕	X	11 101 101 10 100 011	ED A3	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OTIR	(C) ← (HL) B ← B − 1 HL ← HL + 1 Repeat until B = 0	X	1	X	X	X	X	↕	X	11 101 101 10 110 011	ED B3	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTD	(C) ← (HL) B ← B − 1 HL ← HL − 1	X	① ↕	X	X	X	X	↕	X	11 101 101 10 101 011	ED AB	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$

NOTE: ① If the result of B − 1 is zero the Z flag is set, otherwise it is reset.
② N Flag is 1 if data bit is 1, othewise N Flag is 0.

Input and Output Group (Continued)

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210 Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
OTDR	(C) ← (HL) B ← B − 1 HL ← HL − 1 Repeat until B = 0	X	1	X	X	X	X	↕	X	11 101 101 ED 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}

Summary of Flag Operation

Instruction	D_7 S	Z		H		P/V	N	D_0 C	Comments
ADD A, s; ADC A, s	↕	↕	X	↕	X	V	0	↕	8-bit add or add with carry.
SUB s; SBC A, s; CP s; NEG	↕	↕	X	↕	X	V	1	↕	8-bit subtract, subtract with carry, compare and negate accumulator.
AND s	↕	↕	X	1	X	P	0	0	Logical operations.
OR s, XOR s	↕	↕	X	0	X	P	0	0	
INC s	↕	↕	X	↕	X	V	0	•	8-bit increment.
DEC s	↕	↕	X	↕	X	V	1	•	8-bit decrement.
ADD DD, ss	•	•	X	X	X	•	0	↕	16-bit add.
ADC HL, ss	↕	↕	X	X	X	V	0	↕	16-bit add with carry.
SBC HL, ss	↕	↕	X	X	X	V	1	↕	16-bit subtract with carry.
RLA, RLCA, RRA; RRCA	•	•	X	0	X	•	0	↕	Rotate accumulator.
RL m; RLC m; RR m; RRC m; SLA m; SRA m; SRL m	↕	↕	X	0	X	P	0	↕	Rotate and shift locations.
RLD; RRD	↕	↕	X	0	X	P	0	•	Rotate digit left and right.
DAA	↕	↕	X	↕	X	P	•	↕	Decimal adjust accumulator.
CPL	•	•	X	1	X	•	1	•	Complement accumulator.
SCF	•	•	X	0	X	•	0	1	Set carry.
CCF	•	•	X	X	X	•	0	↕	Complement carry.
IN r (C)	↕	↕	X	0	X	P	0	•	Input register indirect.
INI, IND, OUTI; OUTD	X	↕	X	X	X	X	1	•	Block input and output. Z = 0 if B ≠ 0 otherwise Z = 0.
INIR; INDR; OTIR; OTDR	X	1	X	X	X	X	1	•	
LDI; LDD	X	X	X	0	X	↕	0	•	Block transfer instructions. P/V = 1 if BC ≠ 0, otherwise P/V = 0.
LDIR; LDDR	X	X	X	0	X	0	0	•	
CPI; CPIR; CPD; CPDR	X	↕	X	X	X	↕	1	•	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	↕	↕	X	0	X	IFF	0	•	The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag.
BIT b, s	X	↕	X	1	X	X	0	•	The state of bit b of location s is copied into the Z flag.

Symbolic Notation

Symbol	Operation
S	Sign flag. S = 1 if the MSB of the result is 1.
Z	Zero flag. Z = 1 if the result of the operation is 0.
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 the 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 bit 4 of the accumulator.
N	Add/Subtract flag. N = 1 if the previous operation was a subtract.
H & N	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.
C	Carry/Link flag. C = 1 if the operation produced a carry from the MSB of the operand or result.

Symbol	Operation
↕	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 >.

Pin Descriptions

A_0-A_{15}. *Address Bus* (output, active High, 3-state). A_0-A_{15} form a 16-bit address bus. The Address Bus provides the address for memory data bus exchanges (up to 64K bytes) and for I/O device exchanges.

$\overline{\mathrm{BUSACK}}$. *Bus Acknowledge* (output, active Low). Bus Acknowledge indicates to the requesting device that the CPU address bus, data bus, and control signals $\overline{\mathrm{MREQ}}$, $\overline{\mathrm{IORQ}}$, $\overline{\mathrm{RD}}$, and $\overline{\mathrm{WR}}$ have entered their high-impedance states. The external circuitry can now control these lines.

$\overline{\mathrm{BUSREQ}}$. *Bus Request* (input, active Low). Bus Request has a higher priority than $\overline{\mathrm{NMI}}$ and is always recognized at the end of the current machine cycle. $\overline{\mathrm{BUSREQ}}$ forces the CPU address bus, data bus, and control signals $\overline{\mathrm{MREQ}}$, $\overline{\mathrm{IORQ}}$, RD, and WR to go to a high-impedance state so that other devices can control these lines. $\overline{\mathrm{BUSREQ}}$ is normally wire-ORed and requires an external pullup for these applications. Extended $\overline{\mathrm{BUSREQ}}$ periods due to extensive DMA operations can prevent the CPU from properly refreshing dynamic RAMs.

D_0-D_7. *Data Bus* (input/output, active High, 3-state). D_0-D_7 constitute an 8-bit bidirectional data bus, used for data exchanges with memory and I/O.

$\overline{\mathrm{HALT}}$. *Halt State* (output, active Low). $\overline{\mathrm{HALT}}$ indicates that the CPU has executed a Halt instruction and is awaiting either a non-maskable or a maskable interrupt (with the mask enabled) before operation can resume. While halted, the CPU executes NOPs to maintain memory refresh.

$\overline{\mathrm{INT}}$. *Interrupt Request* (input, active Low). Interrupt Request is generated by I/O devices. The CPU honors a request at the end of the current instruction if the internal software-controlled interrupt enable flip-flop (IFF) is enabled. $\overline{\mathrm{INT}}$ is normally wire-ORed and requires an external pullup for these applications.

$\overline{\mathrm{IORQ}}$. *Input/Output Request* (output, active Low, 3-state). $\overline{\mathrm{IORQ}}$ indicates that the lower half of the address bus holds a valid I/O address for an I/O read or write operation. $\overline{\mathrm{IORQ}}$ is also generated concurrently with $\overline{\mathrm{M1}}$ during an interrupt acknowledge cycle to indicate that an interrupt response vector can be placed on the data bus.

$\overline{\mathrm{M1}}$. *Machine Cycle One* (output, active Low). $\overline{\mathrm{M1}}$, together with $\overline{\mathrm{MREQ}}$, indicates that the current machine cycle is the opcode fetch cycle of an instruction execution. M1, together with IORQ, indicates an interrupt acknowledge cycle.

$\overline{\mathrm{MREQ}}$. *Memory Request* (output, active Low, 3-state). $\overline{\mathrm{MREQ}}$ indicates that the address bus holds a valid address for a memory read or memory write operation.

$\overline{\mathrm{NMI}}$. *Non-Maskable Interrupt* (input, negative edge-triggered). $\overline{\mathrm{NMI}}$ has a higher priority than $\overline{\mathrm{INT}}$. $\overline{\mathrm{NMI}}$ is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop, and automatically forces the CPU to restart at location 0066H.

$\overline{\mathrm{RD}}$. *Read* (output, active Low, 3-state). $\overline{\mathrm{RD}}$ indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.

$\overline{\mathrm{RESET}}$. *Reset* (input, active Low). $\overline{\mathrm{RESET}}$ initializes the CPU as follows: it resets the interrupt enable flip-flop, clears the PC and Registers I and R, and sets the interrupt status to Mode 0. During reset time, the address and data bus go to a high-impedance state, and all control output signals go to the inactive state. Note that $\overline{\mathrm{RESET}}$ must be active for a minimum of three full clock cycles before the reset operation is complete.

$\overline{\mathrm{RFSH}}$. *Refresh* (output, active Low). $\overline{\mathrm{RFSH}}$, together with $\overline{\mathrm{MREQ}}$, indicates that the lower seven bits of the system's address bus can be used as a refresh address to the system's dynamic memories.

$\overline{\mathrm{WAIT}}$. *Wait* (input, active Low). $\overline{\mathrm{WAIT}}$ indicates to the CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter a Wait state as long as this signal is active. Extended $\overline{\mathrm{WAIT}}$ periods can prevent the CPU from refreshing dynamic memory properly.

$\overline{\mathrm{WR}}$. *Write* (output, active Low, 3-state). $\overline{\mathrm{WR}}$ indicates that the CPU data bus holds valid data to be stored at the addressed memory or I/O location.

CPU Timing

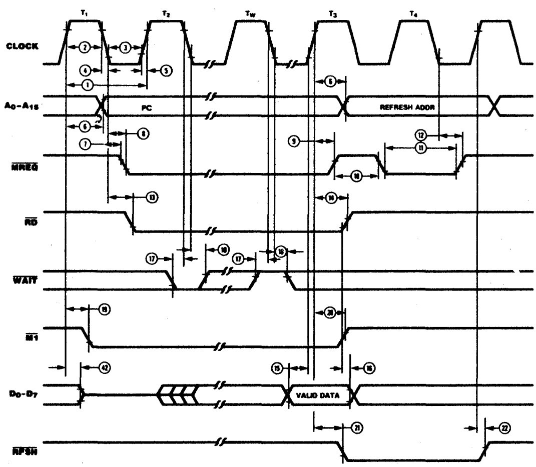
The Z80 CPU executes instructions by proceeding through a specific sequence of operations:

- Memory read or write
- I/O device read or write
- Interrupt acknowledge

The basic clock period is referred to as a T time or cycle, and three or more T cycles make up a machine cycle (M1, M2 or M3 for instance). Machine cycles can be extended either by the CPU automatically inserting one or more Wait states or by the insertion of one or more Wait states by the user.

Instruction Opcode Fetch. The CPU places the contents of the Program Counter (PC) on the address bus at the start of the cycle (Figure 5). Approximately one-half clock cycle later, $\overline{MREQ}$ goes active. When active, $\overline{RD}$ indicates that the memory data can be enabled onto the CPU data bus.

The CPU samples the $\overline{WAIT}$ input with the falling edge of clock state T_2. During clock states T_3 and T_4 of an $\overline{M1}$ cycle dynamic RAM refresh can occur while the CPU starts decoding and executing the instruction. When the Refresh Control signal becomes active, refreshing of dynamic memory can take place.



NOTE: T_W-Wait cycle added when necessary for slow ancilliary devices.

Figure 5. Instruction Opcode Fetch

CPU Timing (Continued)

Memory Read or Write Cycles. Figure 6 shows the timing of memory read or write cycles other than an opcode fetch ($\overline{M1}$) cycle. The $\overline{MREQ}$ and $\overline{RD}$ signals function exactly as in the fetch cycle. In a memory write cycle, $\overline{MREQ}$ also becomes active when the address bus is stable. The $\overline{WR}$ line is active when the data bus is stable, so that it can be used directly as an R/$\overline{W}$ pulse to most semiconductor memories.

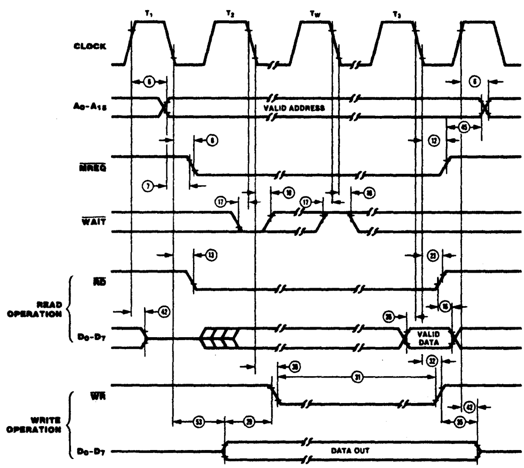
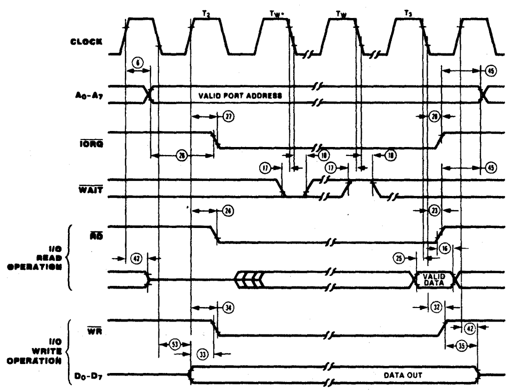


Figure 6. Memory Read or Write Cycles

CPU Timing (Continued)

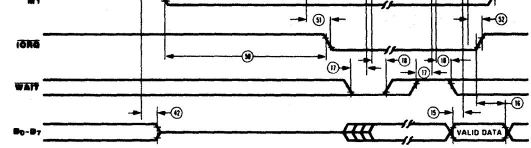
Input or Output Cycles. Figure 7 shows the timing for an I/O read or I/O write operation. During I/O operations, the CPU automatically inserts a single Wait state (T_w). This extra Wait state allows sufficient time for an I/O port to decode the address from the port address lines.



NOTE: T_{W^*} = One Wait cycle automatically inserted by CPU.

Figure 7. Input or Output Cycles

Interrupt Request/Acknowledge Cycle. The CPU samples the interrupt signal with the rising edge of the last clock cycle at the end of any instruction (Figure 8). When an interrupt is accepted, a special $\overline{M1}$ cycle is generated. During this $\overline{M1}$ cycle, $\overline{IORQ}$ becomes active (instead of $\overline{MREQ}$) to indicate that the interrupting device can place an 8-bit vector on the data bus. The CPU automatically adds two Wait states to this cycle.

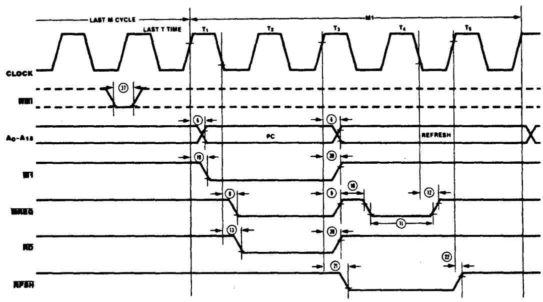


NOTE: 1) T_L = Last state of previous instruction. 2) Two Wait cycles automatically inserted by CPU(*).

Figure 8. Interrupt Request/Acknowledge Cycle

CPU Timing (Continued)

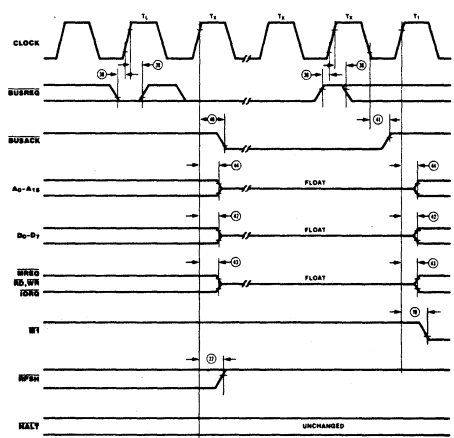
Non-Maskable Interrupt Request Cycle. $\overline{NMI}$ is sampled at the same time as the maskable interrupt input $\overline{INT}$ but has higher priority and cannot be disabled under software control. The subsequent timing is similar to that of a normal instruction fetch except that data put on the bus by the memory is ignored. The CPU instead executes a restart (RST) operation and jumps to the $\overline{NMI}$ service routine located at address 0066H (Figure 9).



*Although $\overline{NMI}$ is an asynchronous input, to guarantee its being recognized on the following machine cycle, $\overline{NMI}$'s falling edge must occur no later than the rising edge of the clock cycle preceding T_{LAST}.

Figure 9. Non-Maskable Interrupt Request Operation

Bus Request/Acknowledge Cycle. The CPU samples $\overline{BUSREQ}$ with the rising edge of the last clock period of any machine cycle (Figure 10). If $\overline{BUSREQ}$ is active, the CPU sets its address, data, and $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$ lines to a high-impedance state with the rising edge of the next clock pulse. At that time, any external device can take control of these lines, usually to transfer data between memory and I/O devices.

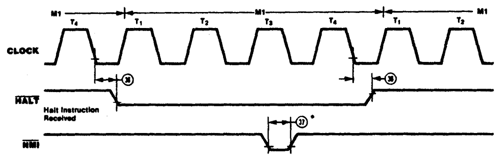


NOTE: T_L = Last state of any M cycle. T_X = An arbitrary clock cycle used by requesting device.

Figure 10. Z-BUS Request/Acknowledge Cycle

CPU Timing (Continued)

Halt Acknowledge Cycle. When the CPU receives a Halt instruction, it executes NOP states until either an $\overline{INT}$ or $\overline{NMI}$ input is received. When in the Halt state, the $\overline{HALT}$ output is active and remains so until an interrupt is received (Figure 11).



NOTE: $\overline{INT}$ will also force a Halt exit.

*See note, Figure 9.

Figure 11. Halt Acknowledge Cycle

Reset Cycle. $\overline{RESET}$ must be active for at least three clock cycles for the CPU to properly accept it. As long as $\overline{RESET}$ remains active, the address and data buses float, and the control outputs are inactive. Once $\overline{RESET}$ goes inactive, three internal T cycles are consumed before the CPU resumes normal processing operation. $\overline{RESET}$ clears the PC register, so the first opcode fetch will be to location 0000 (Figure 12).

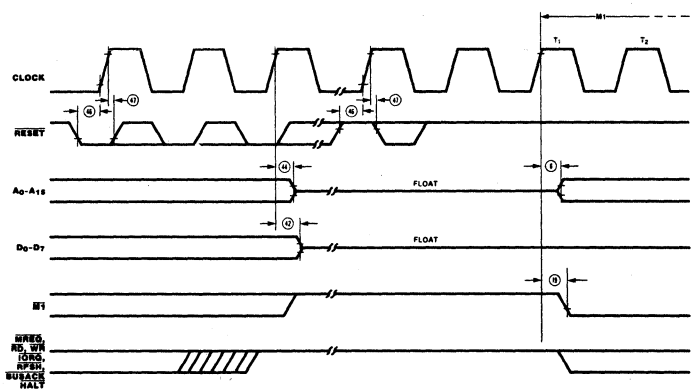


Figure 12. Reset Cycle

AC Characteristics

Number	Symbol	Parameter	Z80 CPU Min	Z80 CPU Max	Z80A CPU Min	Z80A CPU Max	Z80B CPU† Min	Z80B CPU† Max
1	TcC	Clock Cycle Time	400*		250*		165*	
2	TwCh	Clock Pulse Width (High)	180*		110*		65*	
3	TwCl	Clock Pulse Width (Low)	180	2000	110	2000	65	2000
4	TfC	Clock Fall Time	—	30	—	30	—	20
5	TrC	Clock Rise Time		30		30		20
6	TdCr(A)	Clock ↑ to Address Valid Delay	—	145	—	110	—	90
7	TdA(MREQf)	Address Valid to $\overline{MREQ}$ ↓ Delay	125*	—	65*	—	35*	—
8	TdCf(MREQf)	Clock ↓ to $\overline{MREQ}$ ↓ Delay	—	100	—	85	—	70
9	TdCr(MREQr)	Clock ↑ to $\overline{MREQ}$ ↑ Delay	—	100	—	85	—	70
10	TwMREQh	$\overline{MREQ}$ Pulse Width (High)	170*		110*		65*	
11	TwMREQl	$\overline{MREQ}$ Pulse Width (Low)	360*	—	220*	—	135*	—
12	TdCf(MREQr)	Clock ↓ to $\overline{MREQ}$ ↑ Delay	—	100	—	85	—	70
13	TdCf(RDf)	Clock ↓ to $\overline{RD}$ ↓ Delay	—	130	—	95	—	80
14	TdCr(RDr)	Clock ↑ to $\overline{RD}$ ↑ Delay	—	100	—	85	—	70
15	TsD(Cr)	Data Setup Time to Clock ↑	50		35		30	
16	ThD(RDr)	Data Hold Time to $\overline{RD}$ ↑	—	0	—	0	—	0
17	TsWAIT(Cf)	$\overline{WAIT}$ Setup Time to Clock ↓	70	—	70	—	60	—
18	ThWAIT(Cf)	$\overline{WAIT}$ Hold Time after Clock ↓	—	0	—	0	—	0
19	TdCr(M1f)	Clock ↑ to $\overline{M1}$ ↓ Delay	—	130	—	100	—	80
20	TdCr(M1r)	Clock ↑ to $\overline{M1}$ ↑ Delay		130		100		80
21	TdCr(RFSHf)	Clock ↑ to $\overline{RFSH}$ ↓ Delay	—	180	—	130	—	110
22	TdCr(RFSHr)	Clock ↑ to $\overline{RFSH}$ ↑ Delay	—	150	—	120	—	100
23	TdCf(RDr)	Clock ↓ to $\overline{RD}$ ↑ Delay	—	110	—	85	—	70
24	TdCr(RDf)	Clock ↑ to $\overline{RD}$ ↓ Delay	—	100	—	85	—	70
25	TsD(Cf)	Data Setup to Clock ↓ during M_2, M_3, M_4 or M_5 Cycles	60		50		40	
26	TdA(IORQf)	Address Stable prior to $\overline{IORQ}$ ↓	320*	—	180*	—	110*	—
27	TdCr(IORQf)	Clock ↑ to $\overline{IORQ}$ ↓ Delay	—	90	—	75	—	65
28	TdCf(IORQr)	Clock ↓ to $\overline{IORQ}$ ↑ Delay	—	110	—	85	—	70
29	TdD(WRf)	Data Stable prior to $\overline{WR}$ ↓	190*	—	80*	—	25*	—
30	TdCf(WRf)	Clock ↓ to $\overline{WR}$ ↓ Delay		90		80		70
31	TwWR	$\overline{WR}$ Pulse Width	360*	—	220*	—	135*	—
32	TdCf(WRr)	Clock ↓ to $\overline{WR}$ ↑ Delay	—	100	—	80	—	70
33	TdD(WRf)	Data Stable prior to $\overline{WR}$ ↓	20*	—	-10*	—	-55*	—
34	TdCr(WRf)	Clock ↑ to $\overline{WR}$ ↓ Delay	—	80	—	65	—	60
35	TdWRr(D)	Data Stable from $\overline{WR}$ ↑	120*		60*		30*	
36	TdCf(HALT)	Clock ↓ to $\overline{HALT}$ ↑ or ↓	—	300	—	300	—	260
37	TwNMI	$\overline{NMI}$ Pulse Width	80	—	80	—	70	—
38	TsBUSREQ(Cr)	$\overline{BUSREQ}$ Setup Time to Clock ↑	80	—	50	—	50	—

* For clock periods other than the minimums shown in the table, calculate parameters using the expressions in the table on the following page.

† Units in nanoseconds (ns). All timings are preliminary and subject to change.

AC Characteristics (Continued)

Number	Symbol	Parameter	Z80 CPU Min	Z80 CPU Max	Z80A CPU Min	Z80A CPU Max	Z80B CPU† Min	Z80B CPU† Max
39	ThBUSREQ(Cr)	$\overline{BUSREQ}$ Hold Time after Clock ↑	0	—	0	—	0	—
40	TdCr(BUSACKf)	Clock ↑ to $\overline{BUSACK}$ ↓ Delay		120		100		90
41	TdCf(BUSACKr)	Clock ↓ to $\overline{BUSACK}$ ↑ Delay	—	110	—	100	—	90
42	TdCr(Dz)	Clock ↑ to Data Float Delay	—	90	—	90	—	80
43	TdCr(CTz)	Clock ↑ to Control Outputs Float Delay ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$)	—	110	—	80	—	70
44	TdCr(Az)	Clock ↑ to Address Float Delay	—	110	—	90	—	80
45	TdCTr(A)	$\overline{MREQ}$ ↑, $\overline{IORQ}$ ↑, $\overline{RD}$ ↑, and $\overline{WR}$ ↑ to Address Hold Time	160*		80*		35*	
46	TsRESET(Cr)	$\overline{RESET}$ to Clock ↑ Setup Time	90	—	60	—	60	—
47	ThRESET(Cr)	$\overline{RESET}$ to Clock ↑ Hold Time	—	0	—	0	—	0
48	TsINTf(Cr)	$\overline{INT}$ to Clock ↑ Setup Time	80	—	80	—	70	—
49	ThINTr(Cr)	$\overline{INT}$ to Clock ↑ Hold Time	—	0	—	0	—	0
50	TdM1f(IORQf)	$\overline{M1}$ ↓ to $\overline{IORQ}$ ↓ Delay	920*		565*		365*	
51	TdCf(IORQf)	Clock ↓ to $\overline{IORQ}$ ↓ Delay	—	110	—	85	—	70
52	TdCf(IORQr)	Clock ↑ to $\overline{IORQ}$ ↑ Delay	—	100	—	85	—	70
53	TdCf(D)	Clock ↓ to Data Valid Delay	—	230	—	150	—	130

* For clock periods other than the minimums shown in the table, calculate parameters using the following expressions. Calculated values above assumed TrC = TfC = 20 ns.

† Units in nanoseconds (ns). All timings are preliminary and subject to change. All timings assume equal loading on pins with 50 pF.

Footnotes to AC Characteristics

Number	Symbol	Z80	Z80A	Z80B
1	TcC	TwCh + TwCl + TrC + TfC	TwCh + TwCl + TrC + TfC	TwCh + TwCl + TrC + TfC
2	TwCh	Although static by design, TwCh of greater than 200 μs is not guaranteed	Although static by design, TwCh of greater than 200 μs is not guaranteed	Although static by design, TwCh of greater than 200 μs is not guaranteed
7	TdA(MREQf)	TwCh + TfC − 75	TwCh + TfC − 65	TwCh + TfC − 50
10	TwMREQh	TwCh + TfC − 30	TwCh + TfC − 20	TwCh + TfC − 20
11	TwMREQl	TcC − 40	TcC − 30	TcC − 30
26	TdA(IORQf)	TcC − 80	TcC − 70	TcC − 55
29	TdD(WRf)	TcC − 210	TcC − 170	TcC − 140
31	TwWR	TcC − 40	TcC − 30	TcC − 30
33	TdD(WRf)	TwCl + TrC − 180	TwCl + TrC − 140	TwCl + TrC − 140
35	TdWRr(D)	TwCl + TrC − 80	TwCl + TrC − 70	TwCl + TrC − 55
45	TdCTr(A)	TwCl + TrC − 40	TwCl + TrC − 50	TwCl + TrC − 50
50	TdM1f(IORQf)	2TcC + TwCh + TfC − 80	2TcC + TwCh + TfC − 65	2TcC + TwCh + TfC − 50

AC Test Conditions:

V_{IH} = 2.0 V V_{OH} = 2.0 V
V_{IL} = 0.8 V V_{OL} = 0.8 V
V_{IHC} = V_{CC} −0.6 V FLOAT = ±0.5 V
V_{ILC} = 0.45 V

Absolute Maximum Ratings	Storage Temperature .... -65°C to +150°C Temperature under Bias .... Specified operating range Voltages on all inputs and outputs with respect to ground . -0.3 V to +7 V Power Dissipation .... 1.5 W

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

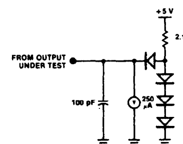
Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature ranges are:

- S* = 0°C to +70°C, +4.75 V ≤ V_{CC} ≤ +5.25 V
- E* = -40°C to +85°C, +4.75 V ≤ V_{CC} ≤ +5.25 V
- M* = -55°C to +125°C, +4.5 V ≤ V_{CC} ≤ +5.5 V

*See Ordering Information section for package temperature range and product number.

All ac parameters assume a load capacitance of 100 pF. Add 10 ns delay for each 50 pF increase in load up to a maximum of 200 pF for the data bus and 100 pF for address and control lines.



DC Characteristics

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-.6	V_{CC}+.3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 1.8 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 µA
I_{CC}	Power Supply Current				
	Z80		150[1]	mA	
	Z80A		200[2]	mA	
	Z80B		200	mA	
I_{LI}	Input Leakage Current		10	µA	V_{IN} = 0 to V_{CC}
I_{LEAK}	3-State Output Leakage Current in Float	-10	10[3]	µA	V_{OUT} = 0.4 to V_{CC}

1. For military grade parts, I_{CC} is 200 mA.
2. Typical rate for Z80A is 90 mA.
3. A_{15}-A_0, D_7-D_0, $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$.

Capacitance

Symbol	Parameter	Min	Max	Unit	Note
C_{CLOCK}	Clock Capacitance		35	pF	
C_{IN}	Input Capacitance		5	pF	Unmeasured pins returned to ground
C_{OUT}	Output Capacitance		10	pF	

T_A = 25°C, f = 1 MHz.

Z8420
Z80® PIO Parallel Input/Output Controller

Zilog

Product Specification

June 1982

Z80 PIO

Features

- Provides a direct interface between Z-80 microcomputer systems and peripheral devices.
- Both ports have interrupt-driven handshake for fast response.
- Four programmable operating modes: byte input, byte output, byte input/output (Port A only), and bit input/output.
- Programmable interrupts on peripheral status conditions.
- Standard Z-80 Family bus-request and prioritized interrupt-request daisy chains implemented without external logic.
- The eight Port B outputs can drive Darlington transistors (1.5 mA at 1.5 V).

General Description

The Z-80 PIO Parallel I/O Circuit is a programmable, dual-port device that provides a TTL-compatible interface between peripheral devices and the Z-80 CPU. The CPU configures the Z-80 PIO to interface with a wide range of peripheral devices with no other external logic. Typical peripheral devices that are compatible with the Z-80 PIO include most keyboards, paper tape readers and punches, printers, PROM programmers, etc.

One characteristic of the Z-80 peripheral controllers that separates them from other interface controllers is that all data transfer between the peripheral device and the CPU is accomplished under interrupt control. Thus, the interrupt logic of the PIO permits full use of the efficient interrupt capabilities of the Z-80 CPU during I/O transfers. All logic necessary to implement a fully nested interrupt structure is included in the PIO.

Another feature of the PIO is the ability to interrupt the CPU upon occurrence of specified status conditions in the peripheral device. For example, the PIO can be programmed to interrupt if any specified peripheral alarm conditions should occur. This interrupt capability reduces the time the processor must spend in polling peripheral status.

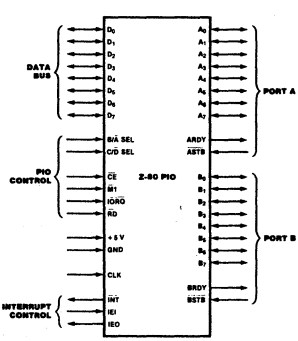


Figure 1. Pin Functions

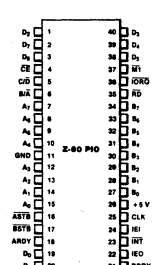


Figure 2. Pin Assignments

General Description (Continued)

The Z-80 PIO interfaces to peripherals via two independent general-purpose I/O ports, designated Port A and Port B. Each port has eight data bits and two handshake signals, Ready and Strobe, which control data transfer. The Ready output indicates to the peripheral that the port is ready for a data transfer. Strobe is an input from the peripheral that indicates when a data transfer has occurred.

Operating Modes. The Z-80 PIO ports can be programmed to operate in four modes: byte output (Mode 0), byte input (Mode 1), byte input/output (Mode 2) and bit input/output (Mode 3).

In Mode 0, either Port A or Port B can be programmed to output data. Both ports have output registers that are individually addressed by the CPU; data can be written to either port at any time. When data is written to a port, an active Ready output indicates to the external device that data is available at the associated port and is ready for transfer to the external device. After the data transfer, the external device responds with an active Strobe input, which generates an interrupt, if enabled.

In Mode 1, either Port A or Port B can be configured in the input mode. Each port has an input register addressed by the CPU. When the CPU reads data from a port, the PIO sets the Ready signal, which is detected by the external device. The external device then places data on the I/O lines and strobes the I/O port, which latches the data into the Port Input Register, resets Ready, and triggers the Interrupt Request, if enabled. The CPU can read the input data at any time, which again sets Ready.

Mode 2 is bidirectional and uses Port A, plus the interrupts and handshake signals from both ports. Port B must be set to Mode 3 and masked off. In operation, Port A is used for both data input and output. Output operation is similar to Mode 0 except that data is allowed out onto the Port A bus only when $\overline{ASTB}$ is Low. For input, operation is similar to Mode 1, except that the data input uses the Port B handshake signals and the Port B interrupt (if enabled).

Both ports can be used in Mode 3. In this mode, the individual bits are defined as either input or output bits. This provides up to eight separate, individually defined bits for each port. During operation, Ready and Strobe are not used. Instead, an interrupt is generated if the condition of one input changes, or if all inputs change. The requirements for generating an interrupt are defined during the programming operation; the active level is specified as either High or Low, and the logic condition is specified as either one input active (OR) or all inputs active (AND). For example, if the port is programmed for active Low inputs and the logic function is AND, then all inputs at the specified port must go Low to generate an interrupt.

Data outputs are controlled by the CPU and can be written or changed at any time.

- Individual bits can be masked off.
- The handshake signals are not used in Mode 3; Ready is held Low, and Strobe is disabled.
- When using the Z-80 PIO interrupts, the Z-80 CPU interrupt mode must be set to Mode 2.

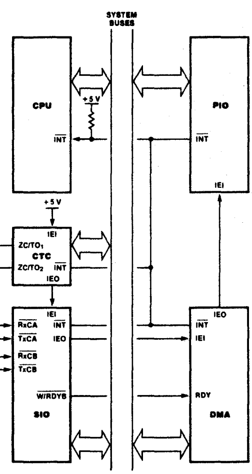


Figure 3. PIO in a Typical Z80 Family Environment

Internal Structure

The internal structure of the Z-80 PIO consists of a Z-80 CPU bus interface, internal control logic, Port A I/O logic, Port B I/O logic, and interrupt control logic (Figure 4). The CPU bus interface logic allows the Z-80 PIO to interface directly to the Z-80 CPU with no other external logic. The internal control logic synchronizes the CPU data bus to the peripheral device interfaces (Port A and Port B). The two I/O ports (A and B) are virtually identical and are used to interface directly to peripheral devices.

Port Logic. Each port contains separate input and output registers, handshake control logic, and the control registers shown in Figure 5. All data transfers between the peripheral unit and the CPU use the data input and output registers. The handshake logic associated with each port controls the data transfers through the input and the output registers. The mode control register (two bits) selects one of the four programmable operating modes.

The control mode (Mode 3) uses the remaining registers. The input/output control register specifies which of the eight data bits in the port are to be outputs and enables these bits; the remaining bits are inputs. The mask register and the mask control register control Mode 3 interrupt conditions. The mask register specifies which of the bits in the port are active and which are masked or inactive.

The mask control register specifies two conditions: first, whether the active state of the input bits is High or Low, and second, whether an interrupt is generated when any *one* unmasked input bit is active (OR condition) or if the interrupt is generated when *all* unmasked input bits are active (AND condition).

Interrupt Control Logic. The interrupt control logic section handles all CPU interrupt protocol for nested-priority interrupt structures. Any device's physical location in a daisy-chain configuration determines its priority. Two lines (IEI and IEO) are provided in each PIO to form this daisy chain. The device closest to the CPU has the highest priority. Within a PIO, Port A interrupts have higher priority than those of Port B. In the byte input, byte output, or bidirectional modes, an interrupt can be generated whenever the peripheral requests a new byte transfer. In the bit control mode, an interrupt can be generated when the peripheral status matches a programmed value. The PIO provides for complete control of nested interrupts. That is, lower priority devices may not interrupt higher priority devices that have not had their interrupt service routines completed by the CPU. Higher priority devices may interrupt the servicing of lower priority devices.

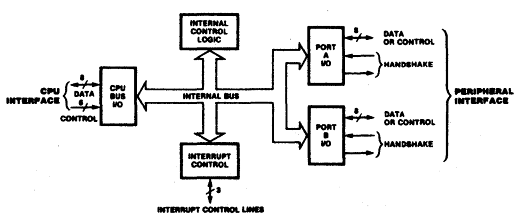


Figure 4. Block Diagram

Internal Structure (Continued)

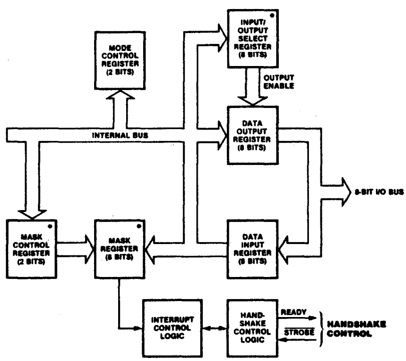
If the CPU (in interrupt Mode 2) accepts an interrupt, the interrupting device must provide an 8-bit interrupt vector for the CPU. This vector forms a pointer to a location in memory where the address of the interrupt service routine is located. The 8-bit vector from the interrupting device forms the least significant eight bits of the indirect pointer while the I Register in the CPU provides the most significant eight bits of the pointer. Each port (A and B) has an independent interrupt vector. The least significant bit of the vector is automatically set to 0 within the PIO because the pointer must point to two adjacent memory locations for a complete 16-bit address.

Unlike the other Z-80 peripherals, the PIO does not enable interrupts immediately after programming. It waits until $\overline{M1}$ goes Low (e.g., during an opcode fetch). This condition is unimportant in the Z-80 environment but might not be if another type of CPU is used.

The PIO decodes the RETI (Return From Interrupt) instruction directly from the CPU data bus so that each PIO in the system knows at all times whether it is being serviced by the CPU interrupt service routine. No other communication with the CPU is required.

CPU Bus I/O Logic. The CPU bus interface logic interfaces the Z-80 PIO directly to the Z-80 CPU, so no external logic is necessary. For large systems, however, address decoders and/or buffers may be necessary.

Internal Control Logic. This logic receives the control words for each port during programming and, in turn, controls the operating functions of the Z-80 PIO. The control logic synchronizes the port operations, controls the port mode, port addressing, selects the read/write function, and issues appropriate commands to the ports and the interrupt logic. The Z-80 PIO does not receive a write input from the CPU; instead, the $\overline{RD}$, $\overline{CE}$, C/$\overline{D}$ and $\overline{IORQ}$ signals generate the write input internally.



*Used in the bit mode only to allow generation of an interrupt if the peripheral I/O pins go to the specified state.

Figure 5. Typical Port I/O Block Diagram

Programming

Mode 0, 1, or 2. *(Byte Input, Output, or Bidirectional).* Programming a port for Mode 0, 1, or 2 requires two words per port. These words are:

A Mode Control Word. Selects the port operating mode (Figure 6). This word may be written any time.

An Interrupt Vector. The Z-80 PIO is designed for use with the Z-80 CPU in interrupt Mode 2 (Figure 7). When interrupts are enabled, the PIO must provide an interrupt vector.

Mode 3. *(Bit Input/Output).* Programming a port for Mode 3 operation requires a control word, a vector (if interrupts are enabled), and three additional words, described as follows:

I/O Register Control. When Mode 3 is selected, the mode control word must be followed by another control word that sets the I/O control register, which in turn defines which port lines are inputs and which are outputs (Figure 8).

Interrupt Control Word. In Mode 3, handshake is not used. Interrupts are generated as a logic function of the input signal levels. The interrupt control word sets the logic conditions and the logic levels required for generating an interrupt. Two logic conditions or functions are available: AND (if all input bits change to the active level, an interrupt is triggered), and OR (if any one of the input bits changes to the active level, an interrupt is triggered). Bit D_6 sets the logic function, as shown in Figure 9. The active level of the input bits can be set either High or Low. The active level is controlled by Bit D_5.

Mask Control Word. This word sets the mask control register, allowing any unused bits to be masked off. If any bits are to be masked, then D_4 must be set. When D_4 is set, the next word written to the port must be a mask control word (Figure 10).

Interrupt Disable. There is one other control word which can be used to enable or disable a port interrupt. It can be used without changing the rest of the interrupt control word (Figure 11).

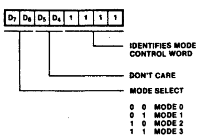


Figure 6. Mode Control Word

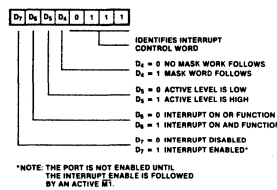


Figure 9. Interrupt Control Word

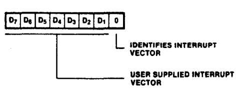


Figure 7. Interrupt Vector Word

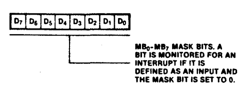


Figure 10. Mask Control Word

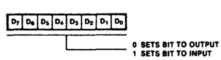


Figure 8. I/O Register Control Word

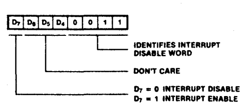


Figure 11. Interrupt Disable Word

Pin Description

A_0-A_7. *Port A Bus* (bidirectional, 3-state). This 8-bit bus transfers data, status, or control information between Port A of the PIO and a peripheral device. A_0 is the least significant bit of the Port A data bus.

ARDY. *Register A Ready* (output, active High). The meaning of this signal depends on the mode of operation selected for Port A as follows:

Output Mode. This signal goes active to indicate that the Port A output register has been loaded and the peripheral data bus is stable and ready for transfer to the peripheral device.

Input Mode. This signal is active when the Port A input register is empty and ready to accept data from the peripheral device.

Bidirectional Mode. This signal is active when data is available in the Port A output register for transfer to the peripheral device. In this mode, data is not placed on the Port A data bus, unless $\overline{\text{ASTB}}$ is active.

Control Mode. This signal is disabled and forced to a Low state.

$\overline{\text{ASTB}}$. *Port A Strobe Pulse From Peripheral Device* (input, active Low). The meaning of this signal depends on the mode of operation selected for Port A as follows:

Output Mode. The positive edge of this strobe is issued by the peripheral to acknowledge the receipt of data made available by the PIO.

Input Mode. The strobe is issued by the peripheral to load data from the peripheral into the Port A input register. Data is loaded into the PIO when this signal is active.

Bidirectional Mode. When this signal is active, data from the Port A output register is gated onto the Port A bidirectional data bus. The positive edge of the strobe acknowledges the receipt of the data.

Control Mode. The strobe is inhibited internally.

B_0-B_7. *Port B Bus* (bidirectional, 3-state). This 8-bit bus transfers data, status, or control information between Port B and a peripheral device. The Port B data bus can supply 1.5 mA at 1.5 V to drive Darlington transistors. B_0 is the least significant bit of the bus.

B/$\overline{\text{A}}$. *Port B Or A Select* (input, High = B). This pin defines which port is accessed during a data transfer between the CPU and the PIO. A Low on this pin selects Port A; a High selects Port B. Often address bit A_0 from the CPU is used for this selection function.

BRDY. *Register B Ready* (output, active High). This signal is similar to ARDY, except that in the Port A bidirectional mode this signal is High when the Port A input register is empty and ready to accept data from the peripheral device.

$\overline{\text{BSTB}}$. *Port B Strobe Pulse From Peripheral Device* (input, active Low). This signal is similar to $\overline{\text{ASTB}}$, except that in the Port A bidirectional mode this signal strobes data from the peripheral device into the Port A input register.

C/$\overline{\text{D}}$. *Control Or Data Select* (input, High = C). This pin defines the type of data transfer to be performed between the CPU and the PIO. A High on this pin during a CPU write to the PIO causes the Z-80 data bus to be interpreted as a *command* for the port selected by the B/$\overline{\text{A}}$ Select line. A Low on this pin means that the Z-80 data bus is being used to transfer data between the CPU and the PIO. Often address bit A_1 from the CPU is used for this function.

$\overline{\text{CE}}$. *Chip Enable* (input, active Low). A Low on this pin enables the PIO to accept command or data inputs from the CPU during a write cycle or to transmit data to the CPU during a read cycle. This signal is generally decoded from four I/O port numbers for Ports A and B, data, and control.

CLK. *System Clock* (input). The Z-80 PIO uses the standard single-phase Z-80 system clock.

D_0-D_7. *Z-80 CPU Data Bus* (bidirectional, 3-state). This bus is used to transfer all data and commands between the Z-80 CPU and the Z-80 PIO. D_0 is the least significant bit.

IEI. *Interrupt Enable In* (input, active High). This signal is used to form a priority-interrupt daisy chain when more than one interrupt-driven device is being used. A High level on this pin indicates that no other devices of higher priority are being serviced by a CPU interrupt service routine.

IEO. *Interrupt Enable Out* (output, active High). The IEO signal is the other signal required to form a daisy chain priority scheme. It is High only if IEI is High and the CPU is not servicing an interrupt from this PIO. Thus this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

$\overline{\text{INT}}$. *Interrupt Request* (output, open drain, active Low). When $\overline{\text{INT}}$ is active the Z-80 PIO is requesting an interrupt from the Z-80 CPU.

$\overline{\text{IORQ}}$. *Input/Output Request* (input from Z-80 CPU, active Low). $\overline{\text{IORQ}}$ is used in conjunction with B/$\overline{\text{A}}$, C/$\overline{\text{D}}$, $\overline{\text{CE}}$, and $\overline{\text{RD}}$ to transfer commands and data between the Z-80 CPU and the Z-80 PIO. When $\overline{\text{CE}}$, $\overline{\text{RD}}$, and $\overline{\text{IORQ}}$ are active, the port addressed by B/$\overline{\text{A}}$ transfers data to the CPU (a read operation). Conversely, when $\overline{\text{CE}}$ and $\overline{\text{IORQ}}$ are active but $\overline{\text{RD}}$ is not, the port addressed by B/$\overline{\text{A}}$ is written into from the CPU with either data or control information, as specified by C/$\overline{\text{D}}$. Also, if $\overline{\text{IORQ}}$ and $\overline{\text{M1}}$ are active simultaneously, the CPU is acknowledging an interrupt; the interrupting port automatically places its interrupt vector on the CPU data bus if it is the highest priority device requesting an interrupt.

Pin Description (Continued)

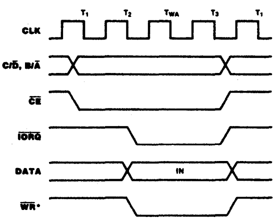
$\overline{\text{M1}}$. *Machine Cycle* (input from CPU, active Low). This signal is used as a sync pulse to control several internal PIO operations. When both the $\overline{\text{M1}}$ and $\overline{\text{RD}}$ signals are active, the Z-80 CPU is fetching an instruction from memory. Conversely, when both $\overline{\text{M1}}$ and $\overline{\text{IORQ}}$ are active, the CPU is acknowledging an interrupt. In addition, $\overline{\text{M1}}$ has two other functions within the Z-80 PIO: it synchronizes the PIO interrupt logic; when $\overline{\text{M1}}$ occurs without an active $\overline{\text{RD}}$ or $\overline{\text{IORQ}}$ signal, the PIO is reset.

$\overline{\text{RD}}$. *Read Cycle Status* (input from Z-80 CPU, active Low). If $\overline{\text{RD}}$ is active, or an I/O operation is in progress, $\overline{\text{RD}}$ is used with B/$\overline{\text{A}}$, C/$\overline{\text{D}}$, $\overline{\text{CE}}$, and $\overline{\text{IORQ}}$ to transfer data from the Z-80 PIO to the Z-80 CPU.

Timing

The following timing diagrams show typical timing in a Z-80 CPU environment. For more precise specifications refer to the composite ac timing diagram.

Write Cycle. Figure 12 illustrates the timing for programming the Z-80 PIO or for writing data to one of its ports. No Wait states are allowed for writing to the PIO other than the automatically inserted T_{WA}. The PIO does not receive a specific write signal; it internally generates its own from the lack of an active $\overline{\text{RD}}$ signal.

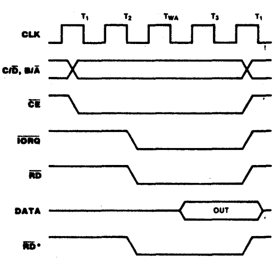


*$\overline{\text{WR}}$ = RD • $\overline{\text{CE}}$ • $\overline{\text{C/D}}$ • $\overline{\text{IORQ}}$

Figure 12. Write Cycle Timing

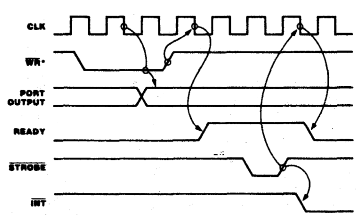
Read Cycle. Figure 13 illustrates the timing for reading the data input from an external device to one of the Z-80 PIO ports. No Wait states are allowed for reading the PIO other than the automatically inserted T_{WA}.

Output Mode (Mode 0). An output cycle (Figure 14) is always started by the execution of an output instruction by the CPU. The $\overline{\text{WR}}$* pulse from the CPU latches the data from the CPU data bus into the selected port's output register. The $\overline{\text{WR}}$* pulse sets the Ready flag after a Low-going edge of CLK, indicating data is available. Ready stays active until the positive edge of the strobe line is received, indicating that data was taken by the peripheral. The positive edge of the strobe pulse generates an $\overline{\text{INT}}$ if the interrupt enable flip-flop has been set and if this device has the highest priority.



*$\overline{\text{RD}}$ = $\overline{\text{RD}}$ • $\overline{\text{CE}}$ • $\overline{\text{C/D}}$ • $\overline{\text{IORQ}}$

Figure 13. Read Cycle Timing

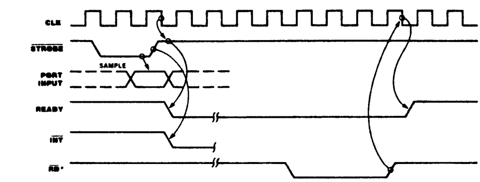


*$\overline{\text{WR}}$ = RD • $\overline{\text{CE}}$ • $\overline{\text{C/D}}$ • $\overline{\text{IORQ}}$

Figure 14. Mode 0 Output Timing

Timing (Continued)

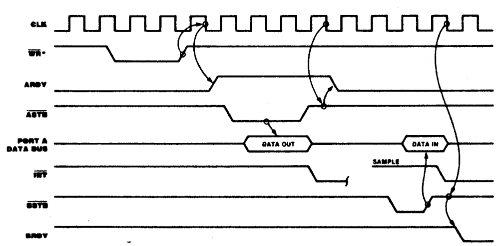
Input Mode (Mode 1). When $\overline{STROBE}$ goes Low, data is loaded into the selected port input register (Figure 15). The next rising edge of strobe activates $\overline{INT}$, if Interrupt Enable is set and this is the highest-priority requesting device. The following falling edge of CLK resets Ready to an inactive state, indicating that the input register is full and cannot accept any more data until the CPU completes a read. When a read is complete, the positive edge of $\overline{RD}$ sets Ready at the next Low-going transition of CLK. At this time new data can be loaded into the PIO.



*$\overline{RD}$ = $\overline{RD}$ • $\overline{CE}$ • $\overline{C/D}$ • $\overline{IORQ}$

Figure 15. Mode 1 Input Timing

Bidirectional Mode (Mode 2). This is a combination of Modes 0 and 1 using all four handshake lines and the eight Port A I/O lines (Figure 16). Port B must be set to the bit mode and its inputs must be masked. The Port A handshake lines are used for output control and the Port B lines are used for input control. If interrupts occur, Port A's vector will be used during port output and Port B's will be used during port input. Data is allowed out onto the Port A bus only when $\overline{ASTB}$ is Low. The rising edge of this strobe can be used to latch the data into the peripheral.



*$\overline{WR}$ = RD • $\overline{CE}$ • $\overline{C/D}$ • $\overline{IORQ}$

Figure 16. Mode 2 Bidirectional Timing

Timing (Continued)

Bit Mode (Mode 3). The bit mode does not utilize the handshake signals, and a normal port write or port read can be executed at any time. When writing, the data is latched into the output registers with the same timing as the output mode (Figure 17).

When reading the PIO, the data returned to the CPU is composed of output register data from those port data lines assigned as outputs and input register data from those port data lines assigned as inputs. The input register contains data that was present immediately prior to the falling edge of $\overline{RD}$. An interrupt is generated if interrupts from the port are enabled and the data on the port data lines satisfy the logical equation defined by the 8-bit mask and 2-bit mask control registers. However, if Port A is programmed in bidirectional mode, Port B does not issue an interrupt in bit mode and must therefore be polled.

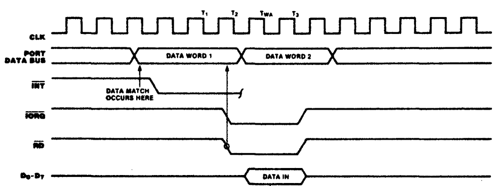


Figure 17. Mode 3 Bit Mode Timing

Interrupt Acknowledge Timing. During $\overline{M1}$ time, peripheral controllers are inhibited from changing their interrupt enable status, permitting the Interrupt Enable signal to ripple through the daisy chain. The peripheral with IEI High and IEO Low during $\overline{INTACK}$ places a preprogrammed 8-bit interrupt vector on the data bus at this time (Figure 18). IEO is held Low until a Return From Interrupt (RETI) instruction is executed by the CPU while IEI is High. The 2-byte RETI instruction is decoded internally by the PIO for this purpose.

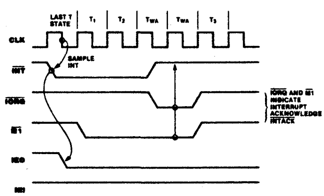


Figure 18. Interrupt Acknowledge Timing

Return From Interrupt Cycle. If a Z-80 peripheral has no interrupt pending and is not under service, then its IEO = IEI. If it has an interrupt under service (i.e., it has already interrupted and received an interrupt acknowledge) then its IEO is always Low, inhibiting lower priority devices from interrupting. If it has an interrupt pending which has not yet been acknowledged, IEO is Low unless an "ED" is decoded as the first byte of a 2-byte opcode (Figure 19). In this case, IEO goes High until the next opcode byte is decoded, whereupon it goes Low again. If the second byte of the opcode was a "4D," then the opcode was an RETI instruction.

After an "ED" opcode is decoded, only the peripheral device which has interrupted and is currently under service has its IEI High and its IEO Low. This device is the highest-priority device in the daisy chain that has received an interrupt acknowledge. All other peripherals have IEI = IEO. If the next opcode byte decoded is "4D," this peripheral device resets its "interrupt under service" condition.

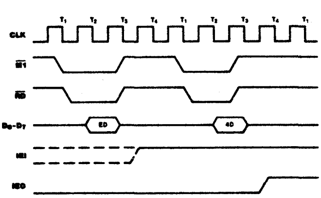
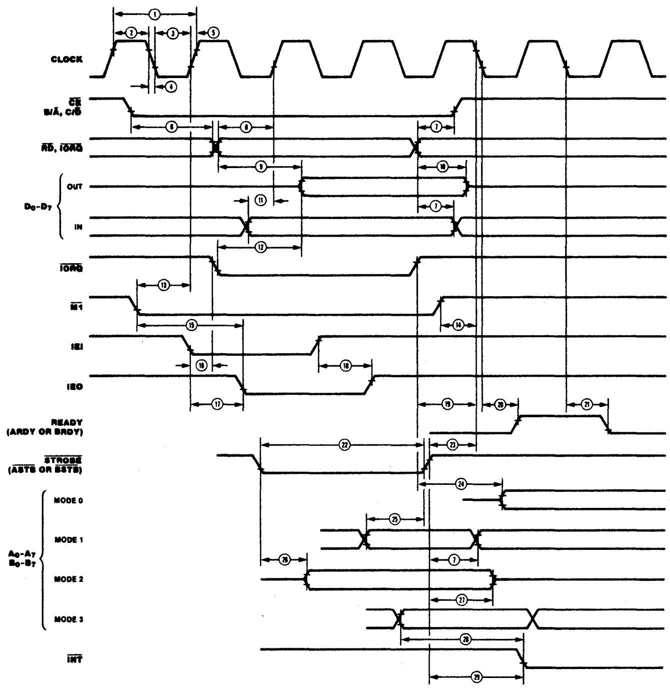


Figure 19. Return From Interrupt

AC Characteristics



Number	Symbol	Parameter	Z-80 PIO Min (ns)	Z-80 PIO Max (ns)	Z-80A PIO Min (ns)	Z-80A PIO Max (ns)	Z-80B PIO[9] Min (ns)	Z-80B PIO[9] Max (ns)	Comment
1	TcC	Clock Cycle Time	400	[1]	250	[1]	165	[1]	
2	TwCh	Clock Width (High)	170	2000	105	2000	65	2000	
3	TwCl	Clock Width (Low)	170	2000	105	2000	65	2000	
4	TfC	Clock Fall Time		30		30		20	
5	TrC	Clock Rise Time		30		30		20	
6	TsCS(RI)	$\overline{\text{CE}}$, B/$\overline{\text{A}}$, C/$\overline{\text{D}}$ to $\overline{\text{RD}}$, $\overline{\text{IORQ}}$ ↓ Setup Time	50		50		50		[6]
7	Th	Any Hold Times for Specified Setup Time	0		0		0	0	
8	TsRI(C)	$\overline{\text{RD}}$, $\overline{\text{IORQ}}$ to Clock ↑ Setup Time	115		115		70		
9	TdRI(DO)	$\overline{\text{RD}}$, $\overline{\text{IORQ}}$ ↓ to Data Out Delay		430		380		300	[2]
10	TdRI(DOs)	$\overline{\text{RD}}$, $\overline{\text{IORQ}}$ ↑ to Data Out Float Delay		160		110		70	
11	TsDI(C)	Data In to Clock ↑ Setup Time	50		50		40		CL = 50 pF
12	TdIO(DOI)	$\overline{\text{IORQ}}$ ↓ to Data Out Delay (INTACK Cycle)	340		160		120		[3]
13	TsM1(Cr)	$\overline{\text{M1}}$ ↓ to Clock ↑ Setup Time	210		90		70		
14	TsM1(Cf)	$\overline{\text{M1}}$ ↑ to Clock ↓ Setup Time ($\overline{\text{M1}}$ Cycle)	0		0		0		[8]
15	TdM1(IEO)	$\overline{\text{M1}}$ ↓ to IEO ↓ Delay (Interrupt Immediately Preceding $\overline{\text{M1}}$ ↓)		300		190		100	[5, 7]
16	TsIEI(IO)	IEI to $\overline{\text{IORQ}}$ ↓ Setup Time (INTACK Cycle)	140		140		100		[7]
17	TdIEI(IEOf)	IEI ↓ to IEO ↓ Delay		190		130		120	[5] CL = 50 pF
18	TdIEI(IEOr)	IEI ↑ to IEO ↑ Delay (after ED Decode)		210		160		160	[5]
19	TcIO(C)	$\overline{\text{IORQ}}$ ↑ to Clock ↓ Setup Time (To Activate READY on Next Clock Cycle)	220		200		170		
20	TdC(RDYr)	Clock ↓ to READY ↑ Delay	200		190		170		[5] CL = 50 pF
21	TdC(RDYf)	Clock ↓ to READY ↑ Delay	150		140		120		[5]
22	TwSTB	$\overline{\text{STROBE}}$ Pulse Width	150		150		120		[4]
23	TsSTB(C)	$\overline{\text{STROBE}}$ ↑ to Clock ↓ Setup Time (To Activate READY on Next Clock Cycle)	220		220		150		[5]
24	TdIO(PD)	$\overline{\text{IORQ}}$ ↑ to PORT DATA Stable Delay (Mode 0)		200		180		160	[5]
25	TsPD(STB)	PORT DATA to $\overline{\text{STROBE}}$ ↑ Setup Time (Mode 1)	260		230		190		
26	TdSTB(PD)	$\overline{\text{STROBE}}$ ↓ to PORT DATA Stable (Mode 2)		230		210		180	[5]
27	TdSTB(PDr)	$\overline{\text{STROBE}}$ ↑ to PORT DATA Float Delay (Mode 2)		200		180		160	CL = 50 pF
28	TdPD(INT)	PORT DATA Match to $\overline{\text{INT}}$ ↓ Delay (Mode 3)		540		490		430	
29	TdSTB(INT)	$\overline{\text{STROBE}}$ ↑ to $\overline{\text{INT}}$ ↓ Delay		490		440		350	

NOTES:
[1] TcC = TwCh + TwCl + TrC + TfC.
[2] Increase TdRI(DO) by 10 ns for each 50 pF increase in load up to 200 pF max.
[3] Increase TdIO(DOI) by 10 ns for each 50 pF, increase in loading up to 200 pF max.
[4] For Mode 2: TwSTB > TsPD(STB).
[5] Increase these values by 2 ns for each 10 pF increase in loading up to 100 pF max.
[6] TsCS(RI) may be reduced. However, the time subtracted from TsCS(RI) will be added to TdRI(DO).
[7] 2.5 TcC > (N-2)TdIEI(IEOf) + TdM1(IEO) + TsIEI(IO) + TTL Buffer Delay, if any.
[8] $\overline{\text{M1}}$ must be active for a minimum of two clock cycles to reset the PIO.
[9] Z80B PIO numbers are preliminary and subject to change.

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . As Specified in Ordering Information

Storage Temperature . . . . -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

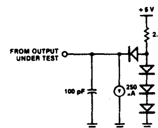
Test Conditions

The characteristics below apply for the following test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature ranges are:

- S* = 0°C to +70°C, +4.75 V ≤ V_{CC} ≤ +5.25 V
- E* = −40°C to +85°C, +4.75 V ≤ V_{CC} ≤ +5.25 V
- M* = −55°C to +125°C, +4.5 V ≤ V_{CC} ≤ +5.5 V

*See Ordering Information section for package temperature range and product number.

All ac parameters assume a load capacitance of 100 pF max. Timing references between two output signals assume a load difference of 50 pF max.



DC Characteristics

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	+0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-0.6	+5.5	V	
V_{IL}	Input Low Voltage	-0.3	+0.8	V	
V_{IH}	Input High Voltage	+2.0	+5.5	V	
V_{OL}	Output Low Voltage		+0.4	V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage	+2.4		V	I_{OH} = -250 μA
I_{LI}	Input Leakage Current	-10.0	+10.0	μA	$0 < V_{IN} < V_{CC}$
I_Z	3-State Output/Data Bus Input Leakage Current	-10.0	+10.0	μA	$0 < V_{IN} < V_{CC}$
I_{CC}	Power Supply Current		100.0	mA	V_{OH} = 1.5V
I_{OHD}	Darlington Drive Current	-1.5	3.8	mA	R_{EXT} = 390 Ω

Over specified temperature and voltage range.

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C	Clock Capacitance		10	pF	Unmeasured pins returned to ground
C_{IN}	Input Capacitance		5	pF	
C_{OUT}	Output Capacitance		10	pF	

Over specified temperature range; f = 1MHz

Z8430
Z80® CTC Counter/ Timer Circuit

Zilog

Product Specification

June 1982

Features

- Four independently programmable counter/timer channels, each with a readable downcounter and a selectable 16 or 256 prescaler. Downcounters are reloaded automatically at zero count.
- Three channels have Zero Count/Timeout outputs capable of driving Darlington transistors.
- Selectable positive or negative trigger initiates timer operation.
- Standard Z-80 Family daisy-chain interrupt structure provides fully vectored, prioritized interrupts without external logic. The CTC may also be used as an interrupt controller.
- Interfaces directly to the Z-80 CPU or—for baud rate generation—to the Z-80 SIO.

General Description

The Z-80 CTC four-channel counter/timer can be programmed by system software for a broad range of counting and timing applications. The four independently programmable channels of the Z-80 CTC satisfy common microcomputer system requirements for event counting, interrupt and interval timing, and general clock rate generation.

System design is simplified because the CTC connects directly to both the Z-80 CPU and the Z-80 SIO with no additional logic. In larger systems, address decoders and buffers may be required.

Programming the CTC is straightforward: each channel is programmed with two bytes; a third is necessary when interrupts are enabled. Once started, the CTC counts down, reloads its time constant automatically, and resumes counting. Software timing loops are completely eliminated. Interrupt processing is simplified because only one vector need be specified; the CTC internally generates a unique vector for each channel.

The Z-80 CTC requires a single +5 V power supply and the standard Z-80 single-phase system clock. It is fabricated with n-channel silicon-gate depletion-load technology, and packaged in a 28-pin plastic or ceramic DIP.

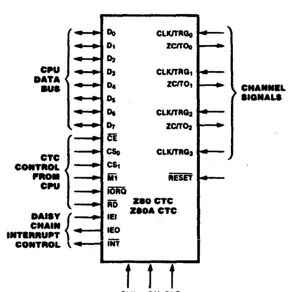


Figure 1. Pin Functions

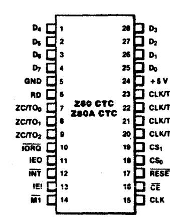


Figure 2. Pin Assignments

Functional Description

The Z-80 CTC has four independent counter/timer channels. Each channel is individually programmed with two words: a control word and a time-constant word. The control word selects the operating mode (counter or timer), enables or disables the channel interrupt, and selects certain other operating parameters. If the timing mode is selected, the control word also sets a prescaler, which divides the system clock by either 16 or 256. The time-constant word is a value from 1 to 256.

During operation, the individual counter channel counts down from the preset time constant value. In counter mode operation the counter decrements on each of the CLK/TRG input pulses until zero count is reached. Each decrement is synchronized by the system clock. For counts greater than 256, more than one counter can be cascaded. At zero count, the down-counter is automatically reset with the time constant value.

The timer mode determines time intervals as small as 4 μs (Z-80A) or 6.4 μs (Z-80) without additional logic or software timing loops. Time intervals are generated by dividing the system clock with a prescaler that decrements a preset down-counter.

Thus, the time interval is an integral multiple of the clock period, the prescaler value (16 or 256) and the time constant that is preset in the down-counter. A timer is triggered automatically when its time constant value is programmed, or by an external CLK/TRG input.

Three channels have two outputs that occur at zero count. The first output is a zero-count/timeout pulse at the ZC/TO output. The fourth channel (Channel 3) does not have a ZC/TO output; interrupt request is the only output available from Channel 3.

The second output is Interrupt Request ($\overline{\text{INT}}$), which occurs if the channel has its interrupt enabled during programming. When the Z-80 CPU acknowledges Interrupt Request, the Z-80 CTC places an interrupt vector on the data bus.

The four channels of the Z-80 CTC are fully prioritized and fit into four contiguous slots in a standard Z-80 daisy-chain interrupt structure. Channel 0 is the highest priority and Channel 3 the lowest. Interrupts can be individually enabled (or disabled) for each of the four channels.

Architecture

The CTC has four major elements, as shown in Figure 3.

- CPU bus I/O
- Channel control logic
- Interrupt logic
- Counter/timer circuits

CPU Bus I/O. The CPU bus I/O circuit decodes the address inputs, and interfaces the CPU data and control signals to the CTC for distribution on the internal bus.

Internal Control Logic. The CTC internal control logic controls overall chip operating functions such as the chip enable, reset, and read/write logic.

Interrupt Logic. The interrupt control logic ensures that the CTC interrupts interface properly with the Z-80 CPU interrupt system. The logic controls the interrupt priority of the CTC as a function of the IEI signal. If IEI is High, the CTC has priority. During interrupt

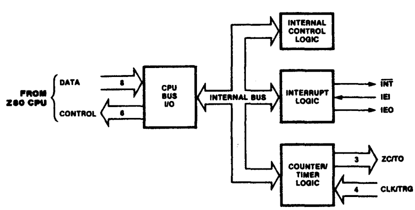


Figure 3. Functional Block Diagram

Architecture (Continued)

processing, the interrupt logic holds IEO Low, which inhibits the interrupt operation on lower priority devices. If the IEI input goes Low, priority is relinquished and the interrupt logic drives IEO Low.

If a channel is programmed to request an interrupt, the interrupt logic drives IEO Low at the zero count, and generates an $\overline{\text{INT}}$ signal to the Z-80 CPU. When the Z-80 CPU responds with interrupt acknowledge ($\overline{\text{M1}}$ and $\overline{\text{IORQ}}$), then the interrupt logic arbitrates the CTC internal priorities, and the interrupt control logic places a unique interrupt vector on the data bus.

If an interrupt is pending, the interrupt logic holds IEO Low. When the Z-80 CPU issues a Return From Interrupt (RETI) instruction, each peripheral device decodes the first byte (ED_{16}). If the device has a pending interrupt, it raises IEO (High) for one $\overline{\text{M1}}$ cycle. This ensures that all lower priority devices can decode the entire RETI instruction and reset properly.

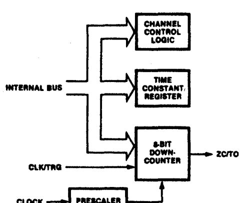


Figure 4. Counter/Timer Block Diagram

Counter/Timer Circuits. The CTC has four independent counter/timer circuits, each containing the logic shown in Figure 4.

Channel Control Logic. The channel control logic receives the 8-bit channel control word when the counter/timer channel is programmed. The channel control logic decodes the control word and sets the following operating conditions:

- Interrupt enable (or disable)
- Operating mode (timer or counter)
- Timer mode prescaler factor (16 or 256)
- Active slope for CLK/TRG input
- Timer mode trigger (automatic or CLK/TRG input)
- Time constant data word to follow
- Software reset

Time Constant Register. When the counter/timer channel is programmed, the time constant register receives and stores an 8-bit time constant value, which can be anywhere from 1 to 256 (0 = 256). This constant is automatically loaded into the down-counter when the counter/timer channel is initialized, and subsequently after each zero count.

Prescaler. The prescaler, which is used only in timer mode, divides the system clock frequency by a factor of either 16 or 256. The prescaler output clocks the down-counter during timer operation. The effect of the prescaler on the down-counter is a multiplication of the system clock period by 16 or 256. The prescaler factor is programmed by bit 5 of the channel control word.

Down-Counter. Prior to each count cycle, the down-counter is loaded with the time constant register contents. The counter is then decremented one of two ways, depending on operating mode:

- By the prescaler output (timer mode)
- By the trigger pulses into the CLK/TRG input (counter mode)

Without disturbing the down-count, the Z-80 CPU can read the count remaining at any time by performing an I/O read operation at the port address assigned to the CTC channel. When the down-counter reaches the zero count, the ZC/TO output generates a positive-going pulse. When the interrupt is enabled, zero count also triggers an interrupt request signal ($\overline{\text{INT}}$) from the interrupt logic.

Programming Each Z-80 CTC channel must be programmed prior to operation. Programming consists of writing two words to the I/O port that corresponds to the desired channel. The first word is a control word that selects the operating mode and other parameters; the second word is a time constant, which is a binary data word with a value from 1 to 256. A time constant word must be preceded by a channel control word.

After initialization, channels may be reprogrammed at any time. If updated control and time constant words are written to a channel during the count operation, the count continues to zero before the new time constant is loaded into the counter.

If the interrupt on any Z-80 CTC channel is enabled, the programming procedure should also include an interrupt vector. Only one vector is required for all four channels, because the interrupt logic automatically modifies the vector for the channel requesting service.

A control word is identified by a 1 in bit 0. A 1 in bit 2 indicates a time constant word is to follow. Interrupt vectors are always addressed to Channel 0, and identified by a 0 in bit 0.

Addressing. During programming, channels are addressed with the channel select pins CS_1 and CS_2. A 2-bit binary code selects the appropriate channel as shown in the following table.

Channel	CS_1	CS_0
0	0	0
1	0	1
2	1	0
3	1	1

Reset. The CTC has both hardware and software resets. The hardware reset terminates all down-counts and disables all CTC interrupts by resetting the interrupt bits in the control registers. In addition, the ZC/TO and Interrupt outputs go inactive, IEO reflects IEI, and D_0-D_7 go to the high-impedance state. All channels must be completely reprogrammed after a hardware reset.

The software reset is controlled by bit 1 in the channel control word. When a channel receives a software reset, it stops counting. When a software reset is used, the other bits in the control word also change the contents of the channel control register. After a software reset a new time constant word must be written to the same channel.

If the channel control word has both bits D_1 and D_2 set to 1, the addressed channel stops operating, pending a new time constant word. The channel is ready to resume after the new constant is programmed. In timer mode, if $D_3 = 0$, operation is triggered automatically when the time constant word is loaded.

Channel Control Word Programming. The channel control word is shown in Figure 5. It sets the modes and parameters described below.

Interrupt Enable. D_7 enables the interrupt, so that an interrupt output ($\overline{INT}$) is generated at zero count. Interrupts may be programmed in either mode and may be enabled or disabled at any time.

Operating Mode. D_6 selects either timer or counter mode.

Prescaler Factor. (Timer Mode Only). D_5 selects factor—either 16 or 256.

Trigger Slope. D_4 selects the active edge or slope of the CLK/TRG input pulses. Note that reprogramming the CLK/TRG slope during operation is equivalent to issuing an active edge. If the trigger slope is changed by a control word update while a channel is pending operation in timer mode, the result is the same as a CLK/TRG pulse and the timer starts. Similarly, if the channel is in counter mode, the counter decrements.

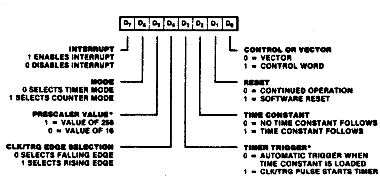


Figure 5. Channel Control Word

Programming (Continued)

Trigger Mode (Timer Mode Only). D_3 selects the trigger mode for timer operation. When D_3 is reset to 0, the timer is triggered automatically. The time constant word is programmed during an I/O write operation, which takes one machine cycle. At the end of the write operation there is a setup delay of one clock period. The timer starts automatically (decrements) on the rising edge of the second clock pulse (T_2) of the machine cycle following the write operation. Once started, the timer runs continuously. At zero count the timer reloads automatically and continues counting without interruption or delay, until stopped by a reset.

When D_3 is set to 1, the timer is triggered externally through the CLK/TRG input. The time constant word is programmed during an I/O write operation, which takes one machine cycle. The timer is ready for operation on the rising edge of the second clock pulse (T_2) of the following machine cycle. Note that the first timer decrement follows the active edge of the CLK/TRG pulse by a delay time of one clock cycle if a minimum setup time to the rising edge of clock is met. If this minimum is not met, the delay is extended by another clock period. Consequently, for immediate triggering, the CLK/TRG input must precede T_2 by one clock cycle plus its minimum setup time. If the minimum time is not met, the timer will start on the third clock cycle (T_3).

Once started the timer operates continuously, without interruption or delay, until stopped by a reset.

Time Constant to Follow. A 1 in D_2 indicates that the next word addressed to the selected channel is a time constant data word for the time constant register. The time constant word may be written at any time.

A 0 in D_2 indicates no time constant word is to follow. This is ordinarily used when the channel is already in operation and the new channel control word is an update. A channel will not operate without a time constant value. The only way to write a time constant value is to write a control word with D_2 set.

Software Reset. Setting D_1 to 1 causes a software reset, which is described in the Reset section.

Control Word. Setting D_0 to 1 identifies the word as a control word.

Time Constant Programming. Before a channel can start counting it must receive a time constant word from the CPU. During programming or reprogramming, a channel control word in which bit 2 is set must precede the time constant word to indicate that the next word is a time constant. The time constant word can be any value from 1 to 256 (Figure 6). Note that 00_{16} is interpreted as 256.

In timer mode, the time interval is controlled by three factors:

- The system clock period (ϕ)
- The prescaler factor (P), which multiplies the interval by either 16 or 256
- The time constant (T), which is programmed into the time constant register

Consequently, the time interval is the product of $\phi \times P \times T$. The minimum timer resolution is $16 \times \phi$ (4 μs with a 4 MHz clock). The maximum timer interval is $256 \times \phi \times 256$ (16.4 ms with a 4 MHz clock). For longer intervals timers may be cascaded.

Interrupt Vector Programming. If the Z-80 CTC has one or more interrupts enabled, it can supply interrupt vectors to the Z-80 CPU. To do so, the Z-80 CTC must be pre-programmed with the most-significant five bits of the interrupt vector. Programming consists of writing a vector word to the I/O port corresponding to the Z-80 CTC Channel 0. Note that D_0 of the vector word is always zero, to distinguish the vector from a channel control word. D_1 and D_2 are not used in programming the vector word. These bits are supplied by the interrupt logic to identify the channel requesting interrupt service with a unique interrupt vector (Figure 7). Channel 0 has the highest priority.

Z80 CTC

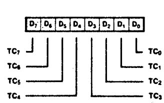


Figure 6. Time Constant Word

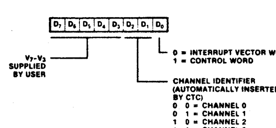


Figure 7. Interrupt Vector Word

Pin Description

$\overline{\textbf{CE}}$. *Chip Enable* (input, active Low). When enabled the CTC accepts control words, interrupt vectors, or time constant data words from the data bus during an I/O write cycle; or transmits the contents of the down-counter to the CPU during an I/O read cycle. In most applications this signal is decoded from the eight least significant bits of the address bus for any of the four I/O port addresses that are mapped to the four counter-timer channels.

CLK. *System Clock* (input). Standard single-phase Z-80 system clock.

CLK/TRG$_0$-CLK/TRG$_3$. *External Clock/Timer Trigger* (input, user-selectable active High or Low). Four pins corresponding to the four Z-80 CTC channels. In counter mode, every active edge on this pin decrements the down-counter. In timer mode, an active edge starts the timer.

CS$_0$-CS$_1$. *Channel Select* (inputs active High). Two-bit binary address code selects one of the four CTC channels for an I/O write or read (usually connected to A_0 and A_1).

D$_0$-D$_7$. *System Data Bus* (bidirectional, 3-state). Transfers all data and commands between the Z-80 CPU and the Z-80 CTC.

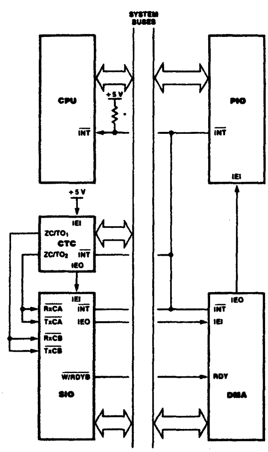


Figure 8. A Typical Z-80 Environment

IEI. *Interrupt Enable In* (input, active High). A High indicates that no other interrupting devices of higher priority in the daisy chain are being serviced by the Z-80 CPU.

IEO. *Interrupt Enable Out* (output, active High). High only if IEI is High and the Z-80 CPU is not servicing an interrupt from any Z-80 CTC channel. IEO blocks lower priority devices from interrupting while a higher priority interrupting device is being serviced.

$\overline{\textbf{INT}}$. *Interrupt Request* (output, open drain, active Low). Low when any Z-80 CTC channel that has been programmed to enable interrupts has a zero-count condition in its down-counter.

$\overline{\textbf{IORQ}}$. *Input/Output Request* (input from CPU, active Low). Used with $\overline{CE}$ and $\overline{RD}$ to transfer data and channel control words between the Z-80 CPU and the Z-80 CTC. During a write cycle, $\overline{IORQ}$ and $\overline{CE}$ are active and $\overline{RD}$ inactive. The Z-80 CTC does not receive a specific write signal; rather, it internally generates its own from the inverse of an active $\overline{RD}$ signal. In a read cycle, $\overline{IORQ}$, $\overline{CE}$ and $\overline{RD}$ are active; the contents of the down-counter are read by the Z-80 CPU. If $\overline{IORQ}$ and $\overline{M1}$ are both true, the CPU is acknowledging an interrupt request, and the highest priority interrupting channel places its interrupt vector on the Z-80 data bus.

$\overline{\textbf{M1}}$. *Machine Cycle One* (input from CPU, active Low). When $\overline{M1}$ and $\overline{IORQ}$ are active, the Z-80 CPU is acknowledging an interrupt. The Z-80 CTC then places an interrupt vector on the data bus if it has highest priority, and if a channel has requested an interrupt ($\overline{INT}$).

$\overline{\textbf{RD}}$. *Read Cycle Status* (input, active Low). Used in conjunction with $\overline{IORQ}$ and $\overline{CE}$ to transfer data and channel control words between the Z-80 CPU and the Z-80 CTC.

$\overline{\textbf{RESET}}$. *Reset* (input active Low). Terminates all down-counts and disables all interrupts by resetting the interrupt bits in all control registers; the ZC/TO and the Interrupt outputs go inactive; IEO reflects IEI; D_0-D_7 go to the high-impedance state.

ZC/TO$_0$-ZC/TO$_2$. *Zero Count/Timeout* (output, active High). Three ZC/TO pins corresponding to Z-80 CTC channels 2 through 0 (Channel 3 has no ZC/TO pin). In both counter and timer modes the output is an active High pulse when the down-counter decrements to zero.

Timing

Read Cycle Timing. Figure 9 shows read cycle timing. This cycle reads the contents of a down-counter without disturbing the count. During clock cycle T_2, the Z-80 CPU initiates a read cycle by driving the following inputs Low: $\overline{RD}$, $\overline{IORQ}$, and $\overline{CE}$. A 2-bit binary code at inputs CS_1 and CS_0 selects the channel to be read. $\overline{M1}$ must be High to distinguish this cycle from an interrupt acknowledge. No additional wait states are allowed.

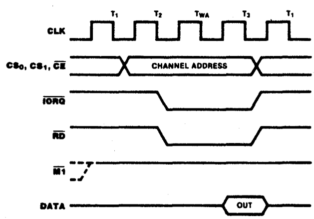


Figure 9. Read Cycle Timing

Write Cycle Timing. Figure 10 shows write cycle timing for loading control, time constant or vector words.

The CTC does not have a write signal input, so it generates one internally when the read ($\overline{RD}$) input is High during T_1. During T_2 $\overline{IORQ}$ and $\overline{CE}$ inputs are Low. $\overline{M1}$ must be High to distinguish a write cycle from an interrupt acknowledge. A 2-bit binary code at inputs CS_1 and CS_0 selects the channel to be addressed, and the word being written is placed on the Z-80 data bus. The data word is latched into the appropriate register with the rising edge of clock cycle T_3.

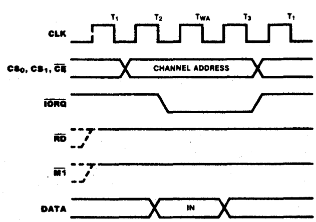


Figure 10. Write Cycle Timing

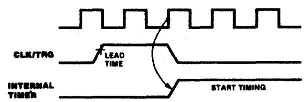


Figure 11. Timer Mode Timing

Timer Operation. In the timer mode, a CLK/TRG pulse input starts the timer (Figure 11) on the second succeeding rising edge of CLK. The trigger pulse is asynchronous, and it must have a minimum width. A minimum lead time (210 ns) is required between the active edge of the CLK/TRG and the next rising edge of CLK to enable the prescaler on the following clock edge. If the CLK/TRG edge occurs closer than this, the initiation of the timer function is delayed one clock cycle. This corresponds to the startup timing discussed in the programming section. The timer can also be started automatically if so programmed by the channel control word.

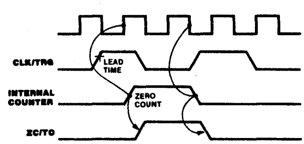


Figure 12. Counter Mode Timing

Counter Operation. In the counter mode, the CLK/TRG pulse input decrements the down-counter. The trigger is asynchronous, but the count is synchronized with CLK. For the decrement to occur on the next rising edge of CLK, the trigger edge must precede CLK by a minimum lead time as shown in Figure 12. If the lead time is less than specified, the count is delayed by one clock cycle. The trigger pulse must have a minimum width, and the trigger period must be at least twice the clock period.

The ZC/TO output occurs immediately after zero count, and follows the rising CLK edge.

Interrupt Operation

The Z-80 CTC follows the Z-80 system interrupt protocol for nested priority interrupts and return from interrupt, wherein the interrupt priority of a peripheral is determined by its location in a daisy chain. Two lines—IEI and IEO—in the CTC connect it to the system daisy chain. The device closest to the +5 V supply has the highest priority (Figure 13). For additional information on the Z-80 interrupt structure, refer to the *Z-80 CPU Product Specification* and the *Z-80 CPU Technical Manual.*

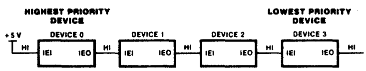


Figure 13. Daisy-Chain Interrupt Priorities

Within the Z-80 CTC, interrupt priority is predetermined by channel number: Channel 0 has the highest priority, and Channel 3 the lowest. If a device or channel is being serviced with an interrupt routine, it cannot be interrupted by a device or channel with lower priority until service is complete. Higher priority devices or channels may interrupt the servicing of lower priority devices or channels.

A Z-80 CTC channel may be programmed to request an interrupt every time its down-counter reaches zero. Note that the CPU must be programmed for interrupt mode 2. Some time after the interrupt request, the CPU sends an interrupt acknowledge. The CTC interrupt control logic determines the highest priority channel that is requesting an interrupt. Then, if the CTC IEI input is High (indicating that it has priority within the system daisy chain) it places an 8-bit interrupt vector on the system data bus. The high-order five bits of this vector ming process; the next two bits are provided by the CTC interrupt control logic as a binary code that identifies the highest priority channel requesting an interrupt; the low-order bit is always zero.

Interrupt Acknowledge Timing. Figure 14 shows interrupt acknowledge timing. After an interrupt request, the Z-80 CPU sends an interrupt acknowledge ($\overline{M1}$ and $\overline{IORQ}$). All channels are inhibited from changing their interrupt request status when $\overline{M1}$ is active—about two clock cycles earlier than $\overline{IORQ}$. $\overline{RD}$ is High to distinguish this cycle from an instruction fetch.

The CTC interrupt logic determines the highest priority channel requesting an interrupt. If the CTC interrupt enable input (IEI) is High, the highest priority interrupting channel within the CTC places its interrupt vector on the data bus when $\overline{IORQ}$ goes Low. Two wait states (T_{WA}) are automatically inserted at this time to allow the daisy chain to stabilize. Additional wait states may be added.

Return from Interrupt Timing. At the end of an interrupt service routine the RETI (Return From Interrupt) instruction initializes the daisy chain enable lines for proper control of nested priority interrupt handling. The CTC decodes the 2-byte RETI code internally and determines whether it is intended for a channel being serviced. Figure 15 shows RETI timing.

If several Z-80 peripherals are in the daisy chain, IEI settles active (High) on the chip currently being serviced when the opcode ED_{16} is decoded. If the following opcode is $4D_{16}$, the peripheral being serviced is released and its IEO becomes active. Additional wait states are allowed.

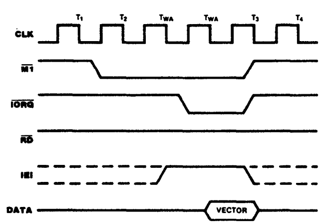


Figure 14. Interrupt Acknowledge Timing

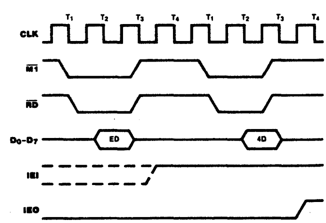


Figure 15. Return From Interrupt Timing

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . As Specified in Ordering Information

Storage Temperature . . . . -65°C to +150°C

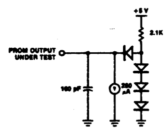
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Test Conditions

The characteristics below apply for the following test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature ranges are:

- S* = 0°C to +70°C, $+4.75\ V \le V_{CC} \le +5.25\ V$
- E* = −40°C to +85°C, $+4.75\ V \le V_{CC} \le +5.25\ V$
- M* = −55°C to +125°C, $+4.5\ V \le V_{CC} \le +5.5\ V$

*See Ordering Information section for package temperature range and product number.



DC Characteristics

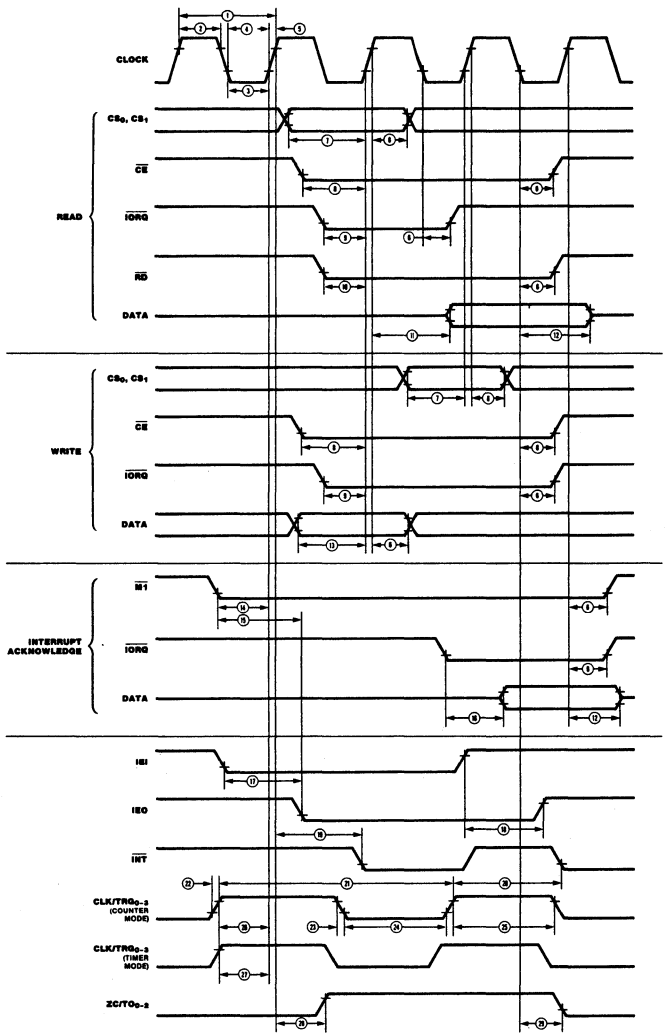
Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	+0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-.6	V_{CC}+.3	V	
V_{IL}	Input Low Voltage	-0.3	+0.8	V	
V_{IH}	Input High Voltage	+2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		+0.4	V	I_{OL} = 2 mA
V_{OH}	Output High Voltage	+2.4		V	I_{OH} = 250 µA
I_{CC}	Power Supply Current		+120	mA	
I_{LI}	Input Leakage Current		+10	µA	V_{IN} = 0 to V_{CC}
I_{LOH}	3-State Output Leakage Current in Float		+10	µA	V_{OUT} = 2.4 to V_{CC}
I_{LOL}	3-State Output Leakage Current in Float		-10	µA	V_{OUT} = 0.4 V
I_{OHD}	Darlington Drive Current	-1.5		mA	V_{OH} = 1.5 V, R_{EXT} = 390Ω

Capacitance

Symbol	Parameter	Max	Unit	Condition
CLK	Clock Capacitance	20	pF	Unmeasured pins returned to ground
C_{IN}	Input Capacitance	5	pF	
C_{OUT}	Output Capacitance	10	pF	

T_A = 25°C, f = 1 MHz

AC Characteristics



Number	Symbol	Parameter	Z-80 CTC Min (ns)	Z-80 CTC Max (ns)	Z-80A CTC Min (ns)	Z-80A CTC Max (ns)	Z-80B CTC Min (ns)	Z-80B CTC Max (ns)	Notes*
1	TcC	Clock Cycle Time	400	[1]	250	[1]	165	[1]	
2	TwCH	Clock Width (High)	170	2000	105	2000	65	2000	
3	TwCl	Clock Width (Low)	170	2000	105	2000	65	2000	
4	TfC	Clock Fall Time		30		30		20	
5	TrC	Clock Rise Time		30		30		20	
6	Th	All Hold Times	0		0		0		
7	TsCS(C)	CS to Clock ↑ Setup Time	250		160		100		
8	TsCE(C)	$\overline{CE}$ to Clock ↑ Setup Time	200		150		100		
9	TsIO(C)	$\overline{IORQ}$ ↓ to Clock ↑ Setup Time	250		115		70		
10	TsRD(C)	$\overline{RD}$ ↓ to Clock ↑ Setup Time	240		115		70		
11	TdC(DO)	Clock ↑ to Data Out Delay		240		200		130	[2]
12	TdC(DOz)	Clock ↓ to Data Out Float Delay		230		110		90	
13	TsDI(C)	Data In to Clock ↑ Setup Time	60		50		40		
14	TsM1(C)	$\overline{M1}$ to Clock ↑ Setup Time	210		90		70		
15	TdM1(IEO)	$\overline{M1}$ ↓ to IEO ↓ Delay (Interrupt immediately preceding $\overline{M1}$)		300		190		130	[3]
16	TdIO(DOI)	$\overline{IORQ}$ ↓ to Data Out Delay (INTA Cycle)		340		160		110	[2]
17	TdIEI(IEOf)	IEI ↓ to IEO ↓ Delay		190		130		100	[3]
18	TdIEI(IEOr)	IEI ↑ to IEO ↑ Delay (After ED Decode)		220		160		110	[3]
19	TdC(INT)	Clock ↑ to $\overline{INT}$ ↓ Delay	(TcC + 200)		(TcC + 140)		TcC + 120		[4]
20	TdCLK(INT)	CLK/TRG ↑ to $\overline{INT}$ ↓							
		tsCTR(C) satisfied	(TcC + 230)		(TcC + 160)		TcC + 130		[5]
		tsCTR(C) not satisfied	(2TcC + 530)		(2TcC + 370)		2TcC + 280		[5]
21	TcCTR	CLK/TRG Cycle Time	(2TcC)		(2TcC)		2TcC		[5]
22	TrCTR	CLK/TRG Rise Time		50		50		40	
23	TfCTR	CLK/TRG Fall Time		50		50		40	
24	TwCTRl	CLK/TRG Width (Low)	200		200		120		
25	TwCTRh	CLK/TRG Width (High)	200		200		120		
26	TsCTR(Cs)	CLK/TRG ↑ to Clock ↑ Setup Time for Immediate Count	300		210		150		[5]
27	TsCTR(Ct)	CLK/TRG ↑ to Clock ↑ Setup Time for enabling of Prescaler on following clock ↑	210		210		150		[4]
28	TdC(ZC/TOr)	Clock ↑ to ZC/TO ↑ Delay		260		190		140	
29	TdC(ZC/TOf)	Clock ↓ to ZC/TO ↓ Delay		190		190		140	

[A] 2.5 TcC > (n-2) TdIEI(IEOf) + TdM1(IEO) + TsIEI(IO) + TTL buffer delay, if any.
[B] $\overline{RESET}$ must be active for a minimum of 3 clock cycles.

NOTES:
[1] TcC = TwCh + TwCl + TrC + TfC.
[2] Increase delay by 10 ns for each 50 pF increase in loading, 200 pF maximum for data lines, and 100 pF for control lines.
[3] Increase delay by 2 ns for each 10 pF increase in loading, 100 pF maximum.
[4] Timer mode.
[5] Counter mode.
[6] RESET must be active for a minimum of 3 clock cycles.
* All timings are preliminary and subject to change.

Z8440
Z80® SIO Serial Input/Output Controller

Zilog

Product Specification

June 1982

Features

- Two independent full-duplex channels, with separate control and status lines for modems or other devices.
- Data rates of 0 to 500K bits/second in the x1 clock mode with a 2.5 MHz clock (Z-80 SIO), or 0 to 800K bits/second with a 4.0 MHz clock (Z-80A SIO).
- Asynchronous protocols: everything necessary for complete messages in 5, 6, 7 or 8 bits/character. Includes variable stop bits and several clock-rate multipliers; break generation and detection; parity; overrun and framing error detection.
- Synchronous protocols: everything necessary for complete bit- or byte-oriented messages in 5, 6, 7 or 8 bits/character, including IBM Bisync, SDLC, HDLC, CCITT-X.25 and others. Automatic CRC generation/checking, sync character and zero insertion/deletion, abort generation/detection and flag insertion.
- Receiver data registers quadruply buffered, transmitter registers doubly buffered.
- Highly sophisticated and flexible daisy-chain interrupt vectoring for interrupts without external logic.

General Description

The Z-80 SIO Serial Input/Output Controller is a dual-channel data communication interface with extraordinary versatility and capability. Its basic functions as a serial-to-parallel, parallel-to-serial converter/controller can be programmed by a CPU for a broad range of serial communication applications.

The device supports all common asynchronous and synchronous protocols, byte- or bit-oriented, and performs all of the functions traditionally done by UARTs, USARTs and synchronous communication controllers combined, plus additional functions traditionally performed by the CPU. Moreover, it does this on two fully-independent channels, with an exceptionally sophisticated interrupt structure that allows very fast transfers.

Full interfacing is provided for CPU or DMA

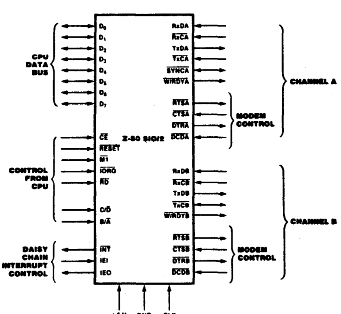


Figure 1. Z-80 SIO/2 Pin Functions

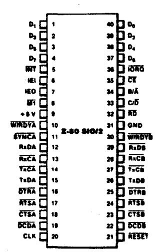


Figure 2. Z-80 SIO/2 Pin Assignments

General Description (Continued)

control. In addition to data communication, the circuit can handle virtually all types of serial I/O with fast (or slow) peripheral devices. While designed primarily as a member of the Z-80 family, its versatility makes it well suited to many other CPUs.

The Z-80 SIO is an n-channel silicon-gate depletion-load device packaged in a 40-pin plastic or ceramic DIP. It uses a single +5 V power supply and the standard Z-80 family single-phase clock.

Pin Description

Figures 1 through 6 illustrate the three pin configurations (bonding options) available in the SIO. The constraints of a 40-pin package make it impossible to bring out the Receive Clock ($\overline{RxC}$), Transmit Clock ($\overline{TxC}$), Data Terminal Ready ($\overline{DTR}$) and Sync ($\overline{SYNC}$) signals for both channels. Therefore, either Channel B lacks a signal or two signals are bonded together in the three bonding options offered:

- Z-80 SIO/2 lacks $\overline{SYNCB}$
- Z-80 SIO/1 lacks $\overline{DTRB}$
- Z-80 SIO/0 has all four signals, but $\overline{TxCB}$ and $\overline{RxCB}$ are bonded together

The first bonding option above (SIO/2) is the preferred version for most applications. The pin descriptions are as follows:

B/$\overline{A}$. *Channel A Or B Select* (input, High selects Channel B). This input defines which channel is accessed during a data transfer between the CPU and the SIO. Address bit A_0 from the CPU is often used for the selection function.

C/$\overline{D}$. *Control Or Data Select* (input, High selects Control). This input defines the type of information transfer performed between the CPU and the SIO. A High at this input during a CPU write to the SIO causes the information on the data bus to be interpreted as a command for the channel selected by B/$\overline{A}$. A Low at C/$\overline{D}$ means that the information on the data bus is data. Address bit A_1 is often used for this function.

$\overline{CE}$. *Chip Enable* (input, active Low). A Low level at this input enables the SIO to accept command or data input from the CPU during write cycle or to transmit data to the CPU during a read cycle.

CLK. *System Clock* (input). The SIO uses the standard Z-80 System Clock to synchronize internal signals. This is a single-phase clock.

$\overline{CTSA}$, $\overline{CTSB}$. *Clear To Send* (inputs, active Low). When programmed as Auto Enables, a Low on these inputs enables the respective transmitter. If not programmed as Auto Enables, these inputs may be programmed as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow risetime signals. The SIO detects pulses on these inputs and interrupts the CPU on both logic level transitions. The Schmitt-trigger bu fering does not guarantee a specified noise-level margin.

D_0-D_7. *System Data Bus* (bidirectional, 3-state). The system data bus transfers data and commands between the CPU and the Z-8 SIO. D_0 is the least significant bit.

$\overline{DCDA}$, $\overline{DCDB}$. *Data Carrier Detect* (inputs, active Low). These pins function as receiver enables if the SIO is programmed for Auto Enables; otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered to accommodate slo risetime signals. The SIO detects pulses on these pins and interrupts the CPU on both logic level transitions. Schmitt-trigger buffer

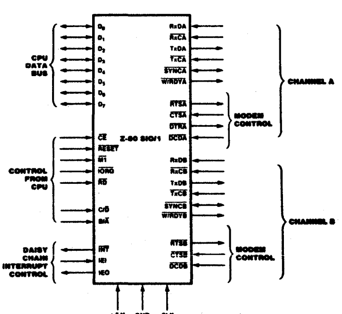


Figure 3. Z-80 SIO/1 Pin Functions

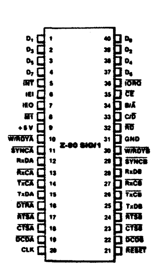


Figure 4. Z-80 SIO/1 Pin Assignments

ing does not guarantee a specific noise-level margin.

$\overline{DTRA}$, $\overline{DTRB}$. *Data Terminal Ready* (outputs, active Low). These outputs follow the state programmed into Z-80 SIO. They can also be programmed as general-purpose outputs.

In the Z-80 SIO/1 bonding option, $\overline{DTRB}$ is omitted.

IEI. *Interrupt Enable In* (input, active High). This signal is used with IEO to form a priority daisy chain when there is more than one interrupt-driven device. A High on this line indicates that no other device of higher priority is being serviced by a CPU interrupt service routine.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from this SIO. Thus, this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

$\overline{INT}$. *Interrupt Request* (output, open drain, active Low). When the SIO is requesting an interrupt, it pulls $\overline{INT}$ Low.

$\overline{IORQ}$. *Input/Output Request* (input from CPU, active Low). $\overline{IORQ}$ is used in conjunction with B/$\overline{A}$, C/$\overline{D}$, $\overline{CE}$ and $\overline{RD}$ to transfer commands and data between the CPU and the SIO. When $\overline{CE}$, $\overline{RD}$ and $\overline{IORQ}$ are all active, the channel selected by B/$\overline{A}$ transfers data to the CPU (a read operation). When $\overline{CE}$ and $\overline{IORQ}$ are active but $\overline{RD}$ is inactive, the channel selected by B/$\overline{A}$ is written to by the CPU with either data or control information as specified by C/$\overline{D}$. If $\overline{IORQ}$ and $\overline{M1}$ are active simultaneously, the CPU is acknowledging an interrupt and the SIO automatically places its interrupt vector on the CPU data bus if it is the highest priority device requesting an interrupt.

$\overline{M1}$. *Machine Cycle* (input from Z-80 CPU, active Low). When $\overline{M1}$ is active and $\overline{RD}$ is also active, the Z-80 CPU is fetching an instruction from memory; when $\overline{M1}$ is active while $\overline{IORQ}$ is active, the SIO accepts $\overline{M1}$ and $\overline{IORQ}$ as an interrupt acknowledge if the SIO is the highest priority device that has interrupted the Z-80 CPU.

$\overline{RxCA}$, $\overline{RxCB}$. *Receiver Clocks* (inputs). Receive data is sampled on the rising edge of $\overline{RxC}$. The Receive Clocks may be 1, 16, 32 or 64 times the data rate in asynchronous modes. These clocks may be driven by the Z-80 CTC Counter Timer Circuit for programmable baud rate generation. Both inputs are Schmitt-trigger buffered (no noise level margin is specified).

In the Z-80 SIO/0 bonding option, $\overline{RxCB}$ is bonded together with $\overline{TxCB}$.

$\overline{RD}$. *Read Cycle Status* (input from CPU, active Low). If $\overline{RD}$ is active, a memory or I/O read operation is in progress. $\overline{RD}$ is used with B/$\overline{A}$, $\overline{CE}$ and $\overline{IORQ}$ to transfer data from the SIO to the CPU.

RxDA, RxDB. *Receive Data* (inputs, active High). Serial data at TTL levels.

$\overline{RESET}$. *Reset* (input, active Low). A Low $\overline{RESET}$ disables both receivers and transmitters, forces TxDA and TxDB marking, forces the modem controls High and disables all interrupts. The control registers must be

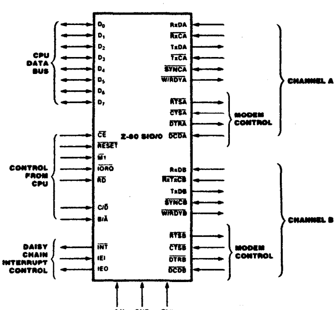


Figure 5. Z-80 SIO/0 Pin Functions

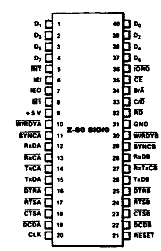


Figure 6. Z-80 SIO/0 Pin Assignments

Pin Description (Continued)

rewritten after the SIO is reset and before data is transmitted or received.

$\overline{\text{RTSA}}$, $\overline{\text{RTSB}}$. *Request To Send* (outputs, active Low). When the RTS bit in Write Register 5 (Figure 14) is set, the $\overline{\text{RTS}}$ output goes Low. When the RTS bit is reset in the Asynchronous mode, the output goes High after the transmitter is empty. In Synchronous modes, the $\overline{\text{RTS}}$ pin strictly follows the state of the RTS bit. Both pins can be used as general-purpose outputs.

$\overline{\text{SYNCA}}$, $\overline{\text{SYNCB}}$. *Synchronization* (inputs/outputs, active Low). These pins can act either as inputs or outputs. In the asynchronous receive mode, they are inputs similar to $\overline{\text{CTS}}$ and $\overline{\text{DCD}}$. In this mode, the transitions on these lines affect the state of the Sync/Hunt status bits in Read Register 0 (Figure 13), but have no other function. In the External Sync mode, these lines also act as inputs. When external synchronization is achieved, $\overline{\text{SYNC}}$ must be driven Low on the second rising edge of $\overline{\text{RxC}}$ after that rising edge of $\overline{\text{RxC}}$ on which the last bit of the sync character was received. In other words, after the sync pattern is detected, the external logic must wait for two full Receive Clock cycles to activate the $\overline{\text{SYNC}}$ input. Once $\overline{\text{SYNC}}$ is forced Low, it should be kept Low until the CPU informs the external synchronization detect logic that synchronization has been lost or a new message is about to start. Character assembly begins on the rising edge of $\overline{\text{RxC}}$ that immediately precedes the falling edge of $\overline{\text{SYNC}}$ in the External Sync mode.

In the internal synchronization mode (Monosync and Bisync), these pins act as outputs that are active during the part of the receive clock ($\overline{\text{RxC}}$) cycle in which sync characters are recognized. The sync condition is not latched, so these outputs are active each time a sync pattern is recognized, regardless of character boundaries.

In the Z-80 SIO/2 bonding option, $\overline{\text{SYNCB}}$ is omitted.

$\overline{\text{TxCA}}$, $\overline{\text{TxCB}}$. *Transmitter Clocks* (inputs). In asynchronous modes, the Transmitter Clocks may be 1, 16, 32 or 64 times the data rate; however, the clock multiplier for the transmitter and the receiver must be the same. The Transmit Clock inputs are Schmitt-trigger buffered for relaxed rise- and fall-time requirements (no noise level margin is specified). Transmitter Clocks may be driven by the Z-80 CTC Counter Timer Circuit for programmable baud rate generation.

In the Z-80 SIO/0 bonding option, $\overline{\text{TxCB}}$ is bonded together with $\overline{\text{RxCB}}$.

TxDA, TxDB. *Transmit Data* (outputs, active High). Serial data at TTL levels. TxD changes from the falling edge of $\overline{\text{TxC}}$.

$\overline{\text{W/RDYA}}$, $\overline{\text{W/RDYB}}$. *Wait/Ready A, Wait/Ready B* (outputs, open drain when programmed for Wait function, driven High and Low when programmed for Ready function). These dual-purpose outputs may be programmed as Ready lines for a DMA controller or as Wait lines that synchronize the CPU to the SIO data rate. The reset state is open drain.

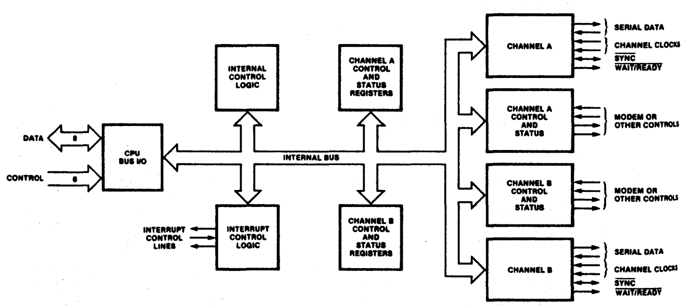


Figure 7. Block Diagram

Functional Description

The functional capabilities of the Z-80 SIO can be described from two different points of view: as a data communications device, it transmits and receives serial data in a wide variety of data-communication protocols; as a Z-80 family peripheral, it interacts with the Z-80 CPU and other peripheral circuits, sharing the data, address and control buses, as well as being a part of the Z-80 interrupt structure. As a peripheral to other microprocessors, the SIO offers valuable features such as non-vectored interrupts, polling and simple handshake capability.

Figure 8 illustrates the conventional devices that the SIO replaces.

The first part of the following discussion covers SIO data-communication capabilities; the second part describes interactions between the CPU and the SIO.

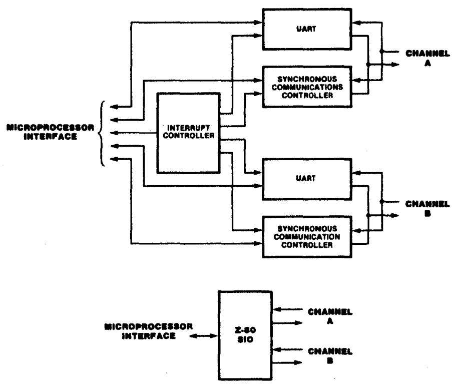


Figure 8. Conventional Devices Replaced by the Z-80 SIO

Data Communication Capabilities

The SIO provides two independent full-duplex channels that can be programmed for use in any common asynchronous or synchronous data-communication protocol. Figure 9 illustrates some of these protocols. The following is a short description of them. A more detailed explanation of these modes can be found in the *Z-80 SIO Technical Manual.*

Asynchronous Modes. Transmission and reception can be done independently on each channel with five to eight bits per character, plus optional even or odd parity. The transmitters can supply one, one-and-a-half or two stop bits per character and can provide a break output at any time. The receiver break-detection logic interrupts the CPU both at the start and end of a received break. Reception is protected from spikes by a transient spike-rejection mechanism that checks the signal one-half a bit time after a Low level is detected on the receive data input (RxDA or RxDB in Figure 5). If the Low does not persist—as in the case of a transient—the character assembly process is not started.

Framing errors and overrun errors are detected and buffered together with the partial character on which they occurred. Vectored interrupts allow fast servicing of error conditions using dedicated routines. Furthermore, a built-in checking process avoids interpreting a framing error as a new start bit: a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit is begun.

The SIO does not require symmetric transmit and receive clock signals—a feature that allows it to be used with a Z-80 CTC or many other clock sources. The transmitter and receiver can handle data at a rate of 1, 1/16, 1/32 or 1/64 of the clock rate supplied to the receive and transmit clock inputs.

In asynchronous modes, the $\overline{\text{SYNC}}$ pin may be programmed as an input that can be used for functions such as monitoring a ring indicator.

Synchronous Modes. The SIO supports both byte-oriented and bit-oriented synchronous communication.

Synchronous byte-oriented protocols can be handled in several modes that allow character synchronization with an 8-bit sync character (Monosync), any 16-bit sync pattern (Bisync), or with an external sync signal. Leading sync

Data Communication Capabilities (Continued)

characters can be removed without interrupting the CPU.

Five-, six- or seven-bit sync characters are detected with 8- or 16-bit patterns in the SIO by overlapping the larger pattern across multiple in-coming sync characters, as shown in Figure 10.

CRC checking for synchronous byte-oriented modes is delayed by one character time so the CPU may disable CRC checking on specific characters. This permits implementation of protocols such as IBM Bisync.

Both CRC-16 ($X^{16} + X^{15} + X^2 + 1$) and CCITT ($X^{16} + X^{12} + X^5 + 1$) error checking polynomials are supported. In all non-SDLC modes, the CRC generator is initialized to 0's; in SDLC modes, it is initialized to 1's. The SIO can be used for interfacing to peripherals such as hard-sectored floppy disk, but it cannot generate or check CRC for IBM-compatible soft-sectored disks. The SIO also provides a feature that automatically transmits CRC data when no other data is available for transmission. This allows very high-speed transmissions under DMA control with no need for CPU intervention at the end of a message. When there is no data or CRC to send in synchronous modes, the transmitter inserts 8- or 16-bit sync characters regardless of the programmed character length.

The SIO supports synchronous bit-oriented protocols such as SDLC and HDLC by performing automatic flag sending, zero insertion and CRC generation. A special command can be used to abort a frame in transmission. At the end of a message the SIO automatically transmits the CRC and trailing flag when the transmit buffer becomes empty. If a transmit underrun occurs in the middle of a message, an external/status interrupt warns the CPU of this status change so that an abort may be issued. One to eight bits per character can be sent, which allows reception of a message with no prior information about the character structure in the information field of a frame.

The receiver automatically synchronizes on the leading flag of a frame in SDLC or HDLC, and provides a synchronization signal on the $\overline{\text{SYNC}}$ pin; an interrupt can also be programmed. The receiver can be programmed to search for frames addressed by a single byte to only a specified user-selected address or to a global broadcast address. In this mode, frames that do not match either the user-selected or broadcast address are ignored. The number of address bytes can be extended under software control. For transmitting data, an interrupt on the first received character or on every character can be selected. The receiver automatically deletes all zeroes inserted by the transmitter during character assembly. It also calculates and automatically checks the CRC to validate frame transmission. At the end of transmission, the status of a received frame is available in the status registers.

The SIO can be conveniently used under DMA control to provide high-speed reception or transmission. In reception, for example, the SIO can interrupt the CPU when the first character of a message is received. The CPU then enables the DMA to transfer the message to memory. The SIO then issues an end-of-frame interrupt and the CPU can check the status of the received message. Thus, the CPU is freed for other service while the message is being received.

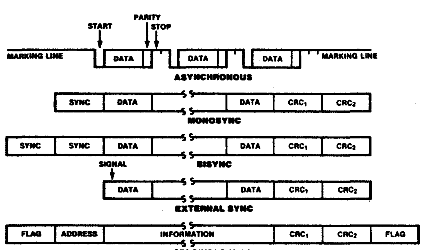


Figure 9. Some Z-80 SIO Protocols

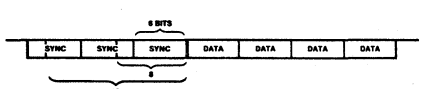


Figure 10.

I/O Interface Capabilities

The SIO offers the choice of polling, interrupt (vectored or non-vectored) and block-transfer modes to transfer data, status and control information to and from the CPU. The block-transfer mode can also be implemented under DMA control.

Polling. Two status registers are updated at appropriate times for each function being performed (for example, CRC error-status valid at the end of a message). When the CPU is operated in a polling fashion, one of the SIO's two status registers is used to indicate whether the SIO has some data or needs some data. Depending on the contents of this register, the CPU will either write data, read data, or just go on. Two bits in the register indicate that a data transfer is needed. In addition, error and other conditions are indicated. The second status register (special receive conditions) does not have to be read in a polling sequence, until a character has been received. All interrupt modes are disabled when operating the device in a polled environment.

Interrupts. The SIO has an elaborate interrupt scheme to provide fast interrupt service in real-time applications. A control register and a status register in Channel B contain the interrupt vector. When programmed to do so, the SIO can modify three bits of the interrupt vector in the status register so that it points directly to one of eight interrupt service routines in memory, thereby servicing conditions in both channels and eliminating most of the needs for a status-analysis routine.

Transmit interrupts, receive interrupts and external/status interrupts are the main sources of interrupts. Each interrupt source is enabled under program control, with Channel A having a higher priority than Channel B, and with receive, transmit and external/status interrupts prioritized in that order within each channel. When the transmit interrupt is enabled, the CPU is interrupted by the transmit buffer becoming empty. (This implies that the transmitter must have had a data character written into it so it can become empty.) The receiver can interrupt the CPU in one of two ways:

- Interrupt on first received character
- Interrupt on all received characters

Interrupt-on-first-received-character is typically used with the block-transfer mode. Interrupt-on-all-received-characters has the option of modifying the interrupt vector in the event of a parity error. Both of these interrupt modes will also interrupt under special receive conditions on a character or message basis (end-of-frame interrupt in SDLC, for example). This means that the special-receive condition can cause an interrupt only if the interrupt-on-first-received-character or interrupt-on-all-received-characters mode is selected. In interrupt-on-first-received-character, an interrupt can occur from special-receive conditions (except parity error) after the first-received-character interrupt (example: receive-overrun interrupt).

The main function of the external/status interrupt is to monitor the signal transitions of the Clear To Send ($\overline{CTS}$), Data Carrier Detect ($\overline{DCD}$) and Synchronization ($\overline{SYNC}$) pins (Figures 1 through 6). In addition, an external/status interrupt is also caused by a CRC-sending condition or by the detection of a break sequence (asynchronous mode) or abort sequence (SDLC mode) in the data stream. The interrupt caused by the break/abort sequence allows the SIO to interrupt when the break/abort sequence is detected or terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the break/abort condition in external logic.

I/O Interface Capabilities (Continued)

In a Z-80 CPU environment (Figure 11), SIO interrupt vectoring is "automatic": the SIO passes its internally-modifiable 8-bit interrupt vector to the CPU, which adds an additional 8 bits from its interrupt-vector (I) register to form the memory address of the interrupt-routine table. This table contains the address of the beginning of the interrupt routine itself. The process entails an indirect transfer of CPU control to the interrupt routine, so that the next instruction executed after an interrupt acknowledge by the CPU is the first instruction of the interrupt routine itself.

CPU/DMA Block Transfer. The SIO's block-transfer mode accommodates both CPU block transfers and DMA controllers (Z-80 DMA or other designs). The block-transfer mode uses the Wait/Ready output signal, which is selected with three bits in an internal control register. The Wait/Ready output signal can be programmed as a $\overline{\text{WAIT}}$ line in the CPU block-transfer mode or as a $\overline{\text{READY}}$ line in the DMA block-transfer mode.

To a DMA controller, the SIO $\overline{\text{READY}}$ output indicates that the SIO is ready to transfer data to or from memory. To the CPU, the $\overline{\text{WAIT}}$ output indicates that the SIO is not ready to transfer data, thereby requesting the CPU to extend the I/O cycle.

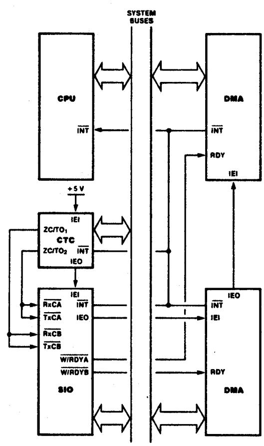


Figure 11. Typical Z-80 Environment

Internal Structure

The internal structure of the device includes a Z-80 CFU interface, internal control and interrupt logic, and two full-duplex channels. Each channel contains its own set of control and status (write and read) registers, and control and status logic that provides the interface to modems or other external devices.

The registers for each channel are designated as follows:

WR0-WR7 — Write Registers 0 through 7
RR0-RR2 — Read Registers 0 through 2

The register group includes five 8-bit control registers, two sync-character registers and two status registers. The interrupt vector is written into an additional 8-bit register (Write Register 2) in Channel B that may be read through another 8-bit register (Read Register 2) in Channel B. The bit assignment and functional grouping of each register is configured to simplify and organize the programming process. Table 1 lists the functions assigned to each read or write register.

Read Register Functions	
RR0	Transmit/Receive buffer status, interrupt status and external status
RR1	Special Receive Condition status
RR2	Modified interrupt vector (Channel B only)

Write Register Functions	
WR0	Register pointers, CRC initialize, initialization commands for the various modes, etc.
WR1	Transmit/Receive interrupt and data transfer mode definition.
WR2	Interrupt vector (Channel B only)
WR3	Receive parameters and control
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls
WR6	Sync character or SDLC address field
WR7	Sync character or SDLC flag

Internal Structure (Continued)

The logic for both channels provides formats, synchronization and validation for data transferred to and from the channel interface. The modem control inputs, Clear To Send ($\overline{CTS}$) and Data Carrier Detect ($\overline{DCD}$), are monitored by the external control and status logic under program control. All external control-and-status-logic signals are general-purpose in nature and can be used for functions other than modem control.

Data Path. The transmit and receive data path illustrated for Channel A in Figure 12 is identical for both channels. The receiver has three 8-bit buffer registers in a FIFO arrangement, in addition to the 8-bit receive shift register. This scheme creates additional time for the CPU to service an interrupt at the beginning of a block of high-speed data. Incoming data is routed through one of several paths (data or CRC) depending on the selected mode and—in asynchronous modes—the character length.

The transmitter has an 8-bit transmit data buffer register that is loaded from the internal data bus, and a 20-bit transmit shift register that can be loaded from the sync-character buffers or from the transmit data register. Depending on the operational mode, outgoing data is routed through one of four main paths before it is transmitted from the Transmit Data output (TxD).

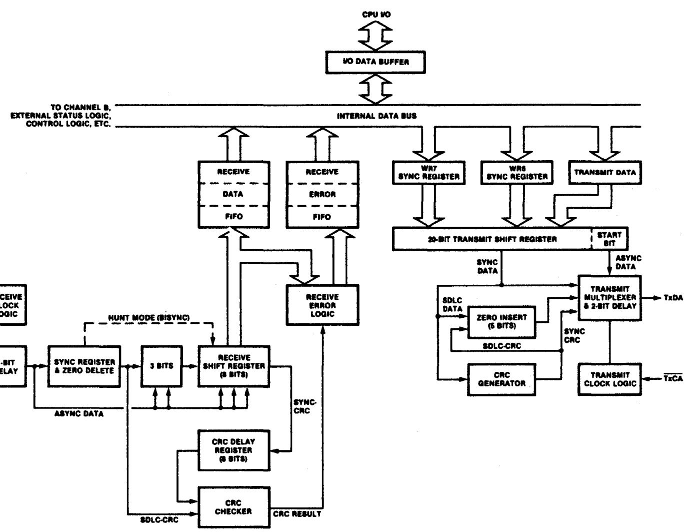


Figure 12. Transmit and Receive Data Path (Channel A)

Programming

The system program first issues a series of commands that initialize the basic mode of operation and then other commands that qualify conditions within the selected mode. For example, the asynchronous mode, character length, clock rate, number of stop bits, even or odd parity might be set first; then the interrupt mode; and finally, receiver or transmitter enable.

Both channels contain registers that must be programmed via the system program prior to operation. The channel-select input ($B/\overline{A}$) and the control/data input ($C/\overline{D}$) are the command-structure addressing controls, and are normally controlled by the CPU address bus. Figures 15 and 16 illustrate the timing relationships for programming the write registers and transferring data and status.

Read Registers. The SIO contains three read registers for Channel B and two read registers for Channel A (RR0-RR2 in Figure 13) that can be read to obtain the status information; RR2 contains the internally-modifiable interrupt vector and is only in the Channel B register set. The status information includes error conditions, interrupt vector and standard communications-interface signals.

To read the contents of a selected read register other than RR0, the system program must first write the pointer byte to WR0 in exactly the same way as a write register operation. Then, by executing a read instruction, the contents of the addressed read register can be read by the CPU.

The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring. For example, when the interrupt vector indicates that a Special Receive Condition interrupt has occurred, all the appropriate error bits can be read from a single register (RR1).

Write Registers. The SIO contains eight write registers for Channel B and seven write registers for Channel A (WR0-WR7 in Figure 14) that are programmed separately to configure the functional personality of the channels; WR2 contains the interrupt vector for both channels and is only in the Channel B register set. With the exception of WR0, programming the write registers requires two bytes. The first byte is to WR0 and contains three bits (D_0-D_2) that point to the selected register; the second byte is the actual control word that is written into the register to configure the SIO.

WR0 is a special case in that all of the basic commands can be written to it with a single byte. Reset (internal or external) initializes the pointer bits D_0-D_2 to point to WR0. This implies that a channel reset must not be combined with the pointing to any register.

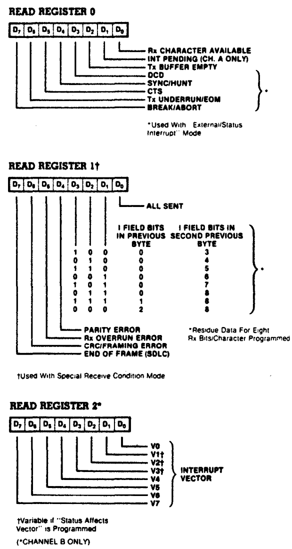


Figure 13. Read Register Bit Functions

Programming (Continued)

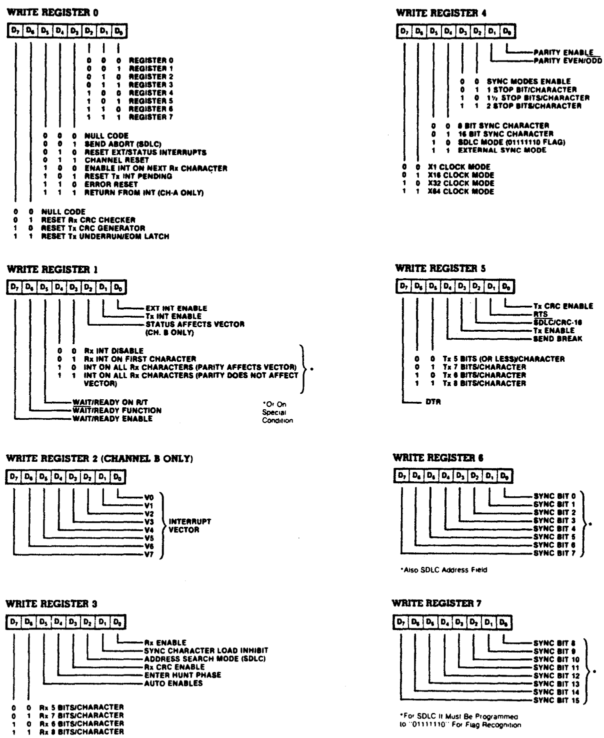


Figure 14. Write Register Bit Functions

Timing

The SIO must have the same clock as the CPU (same phase and frequency relationship, not necessarily the same driver).

Read Cycle. The timing signals generated by a Z-80 CPU input instruction to read a data or status byte from the SIO are illustrated in Figure 15.

Write Cycle. Figure 16 illustrates the timing and data signals generated by a Z-80 CPU output instruction to write a data or control byte into the SIO.

Interrupt-Acknowledge Cycle. After receiving an interrupt-request signal from an SIO ($\overline{INT}$ pulled Low), the Z-80 CPU sends an interrupt-acknowledge sequence ($\overline{M1}$ Low, and $\overline{IORQ}$ Low a few cycles later) as in Figure 17.

The SIO contains an internal daisy-chained interrupt structure for prioritizing nested interrupts for the various functions of its two channels, and this structure can be used within an external user-defined daisy chain that prioritizes several peripheral circuits.

The IEI of the highest-priority device is terminated High. A device that has an interrupt pending or under service forces its IEO Low. For devices with no interrupt pending or under service, IEO = IEI.

To insure stable conditions in the daisy chain, all interrupt status signals are prevented from changing while $\overline{M1}$ is Low. When $\overline{IORQ}$ is Low, the highest priority interrupt requestor (the one with IEI High) places its interrupt vector on the data bus and sets its internal interrupt-under-service latch.

Return From Interrupt Cycle. Figure 18 illustrates the return from interrupt cycle. Normally, the Z-80 CPU issues a RETI (Return From Interrupt) instruction at the end of an interrupt service routine. RETI is a 2-byte opcode (ED-4D) that resets the interrupt-under-service latch in the SIO to terminate the interrupt that has just been processed. This is accomplished by manipulating the daisy chain in the following way.

The normal daisy-chain operation can be used to detect a pending interrupt; however, it cannot distinguish between an interrupt under service and a pending unacknowledged interrupt of a higher priority. Whenever "ED" is decoded, the daisy chain is modified by forcing High the IEO of any interrupt that has not yet been acknowledged. Thus the daisy chain identifies the device presently under service as the only one with an IEI High and an IEO Low. If the next opcode byte is "4D," the interrupt-under-service latch is reset.

The ripple time of the interrupt daisy chain (both the High-to-Low and the Low-to-High transitions) limits the number of devices that can be placed in the daisy chain. Ripple time can be improved with carry-look-ahead, or by extending the interrupt-acknowledge cycle. For further information about techniques for increasing the number of daisy-chained devices, refer to the *Z-80 CPU Product Specification.*

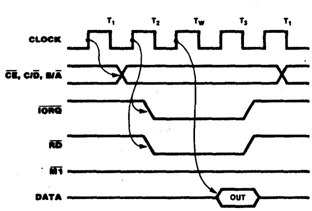


Figure 15. Read Cycle

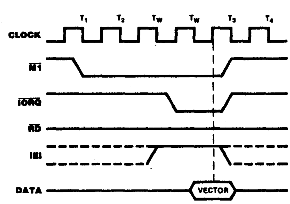


Figure 17. Interrupt Acknowledge Cycle

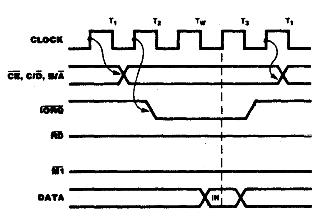


Figure 16. Write Cycle

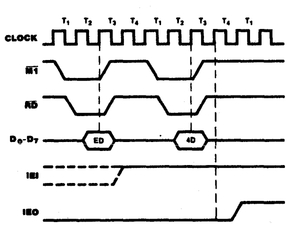


Figure 18. Return from Interrupt Cycle

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . As Specified in Ordering Information

Storage Temperature . . . . -65°C to +150°C

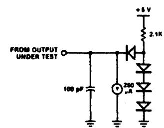
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Test Conditions

The characteristics below apply for the following test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature ranges are:

- S* = 0°C to +70°C, $+4.75\text{ V} \le V_{CC} \le +5.25\text{ V}$
- E* = −40°C to +85°C, $+4.75\text{ V} \le V_{CC} \le +5.25\text{ V}$
- M* = −55°C to +125°C, $+4.5\text{ V} \le V_{CC} \le +5.5\text{ V}$

*See Ordering Information section for package temperature range and product number.



DC Characteristics

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	+0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-0.6	+5.5	V	
V_{IL}	Input Low Voltage	-0.3	+0.8	V	
V_{IH}	Input High Voltage	+2.0	+5.5	V	
V_{OL}	Output Low Voltage		+0.4	V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage	+2.4		V	I_{OH} = -250 μA
I_{LI}	Input Leakage Current	-10	+10	μA	$0<V_{IN}<V_{CC}$
I_Z	3-State Output/Data Bus Input Leakage Current	-10	+10	μA	$0<V_{IN}<V_{CC}$
$I_{L(SY)}$	$\overline{\text{SYNC}}$ Pin Leakage Current	-40	+10	μA	$0<V_{IN}<V_{CC}$
I_{CC}	Power Supply Current		100	mA	

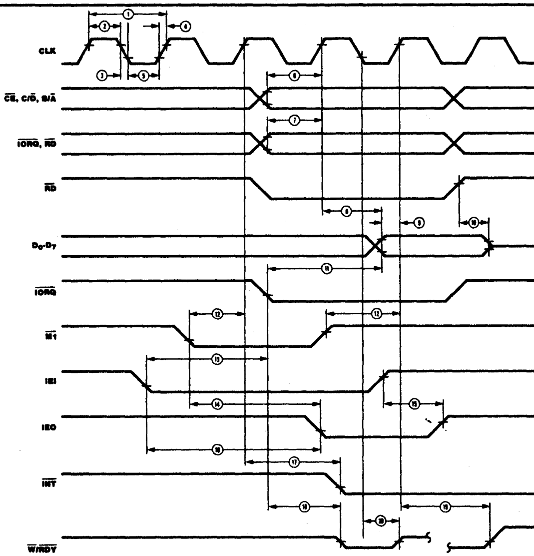
Over specified temperature and voltage range.

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C	Clock Capacitance		40	pF	Unmeasured
C_{IN}	Input Capacitance		5	pF	pins returned
C_{OUT}	Output Capacitance		10	pF	to ground

Over specified temperature range; f = 1MHz

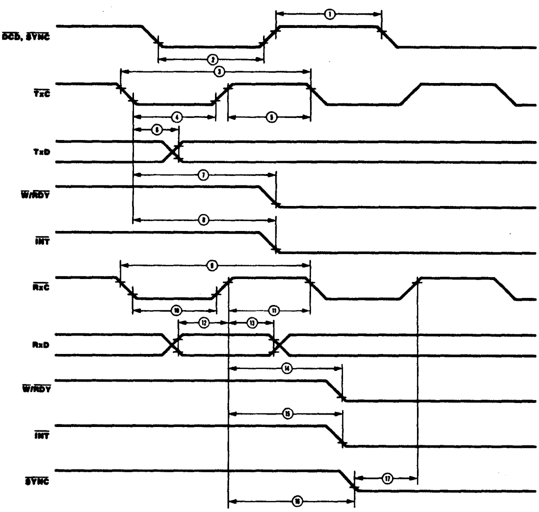
AC Electrical Characteristics



Number	Symbol	Parameter	Z-80 SIO Min	Z-80 SIO Max	Z-80A SIO Min	Z-80A SIO Max	Z-80B SIO*† Min	Z-80B SIO*† Max
1	TcC	Clock Cycle Time	400	4000	250	4000	165	4000
2	TwCh	Clock Width (High)	170	2000	105	2000	70	2000
3	TfC	Clock Fall Time		30		30		15
4	TrC	Clock Rise Time		30		30		15
5	TwCl	Clock Width (Low)	170	2000	105	2000	70	2000
6	TsAD(C)	$\overline{CE}$, C/$\overline{D}$, B/$\overline{A}$ to Clock ↑ Setup Time	160		145		60	
7	TsCS(C)	$\overline{IORQ}$, $\overline{RD}$ to Clock ↑ Setup Time	240		115		60	
8	TdC(DO)	Clock ↑ to Data Out Delay		240		220		150
9	TsDI(C)	Data In to Clock ↑ Setup (Write or $\overline{M1}$ Cycle)	50		50		30	
10	TdRD(DOz)	$\overline{RD}$ ↑ to Data Out Float Delay		230		110		90
11	TdIO(DOI)	$\overline{IORQ}$ ↓ to Data Out Delay (INTACK Cycle)		340		160		100
12	TsM1(C)	$\overline{M1}$ to Clock ↑ Setup Time	210		90		75	
13	TsIEI(IO)	IEI to $\overline{IORQ}$ ↓ Setup Time (INTACK Cycle)	200		140		120	
14	TdM1(IEO)	$\overline{M1}$ ↓ to IEO ↓ Delay (interrupt before $\overline{M1}$)		300		190	160	
15	TdIEI(IEOr)	IEI ↑ to IEO ↑ Delay (after ED decode)		150		100		70
16	TdIEI(IEOf)	IEI ↓ to IEO ↓ Delay		150		100		70
17	TdC(INT)	Clock ↑ to $\overline{INT}$ ↓ Delay		200		200		150
18	TdIO(W/RWf)	$\overline{IORQ}$ ↓ or $\overline{CE}$ ↓ to W/$\overline{RDY}$ ↓ (Delay Wait Mode)		300		210		175
19	TdC(W/RR)	Clock ↑ to W/$\overline{RDY}$ ↑ Delay (Ready Mode)		120		120		100
20	TdC(W/RWz)	Clock ↓ to W/$\overline{RDY}$ Float Delay (Wait Mode)		150		130		110
21	Th	Any unspecified Hold when Setup is specified	0		0		0	

* Z-80 SIO timings are preliminary and subject to change.
† Units in nanoseconds (ns).

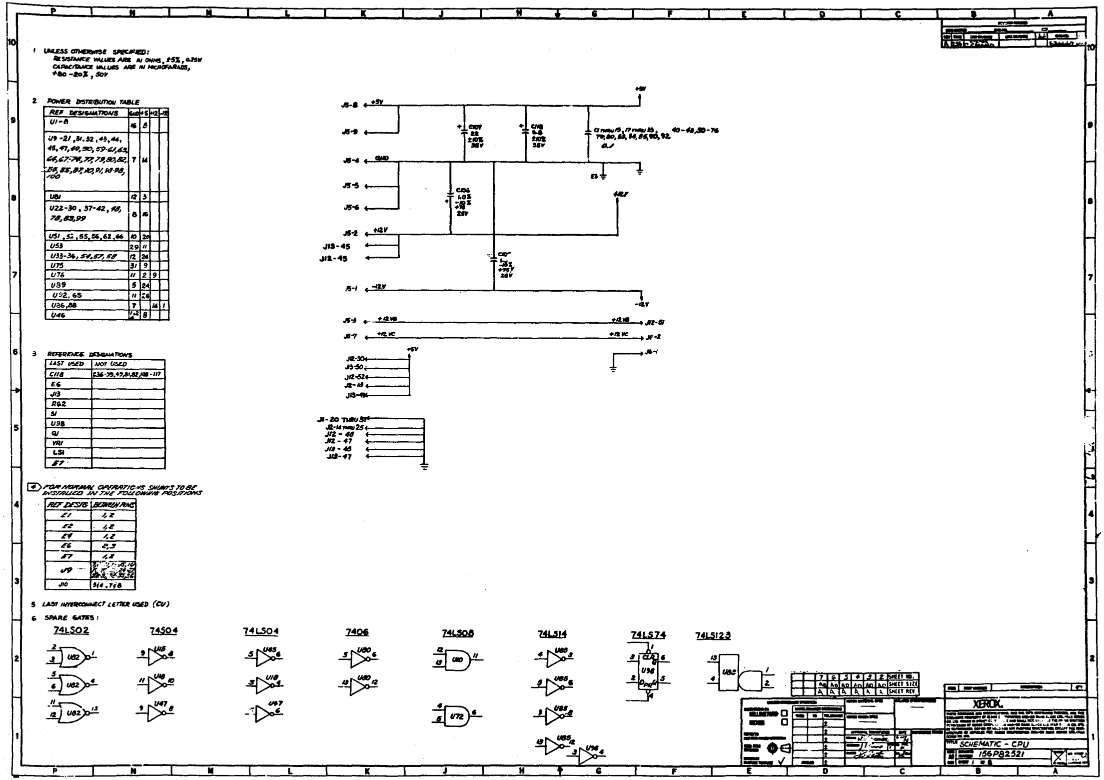
AC Electrical Characteristics (Continued)

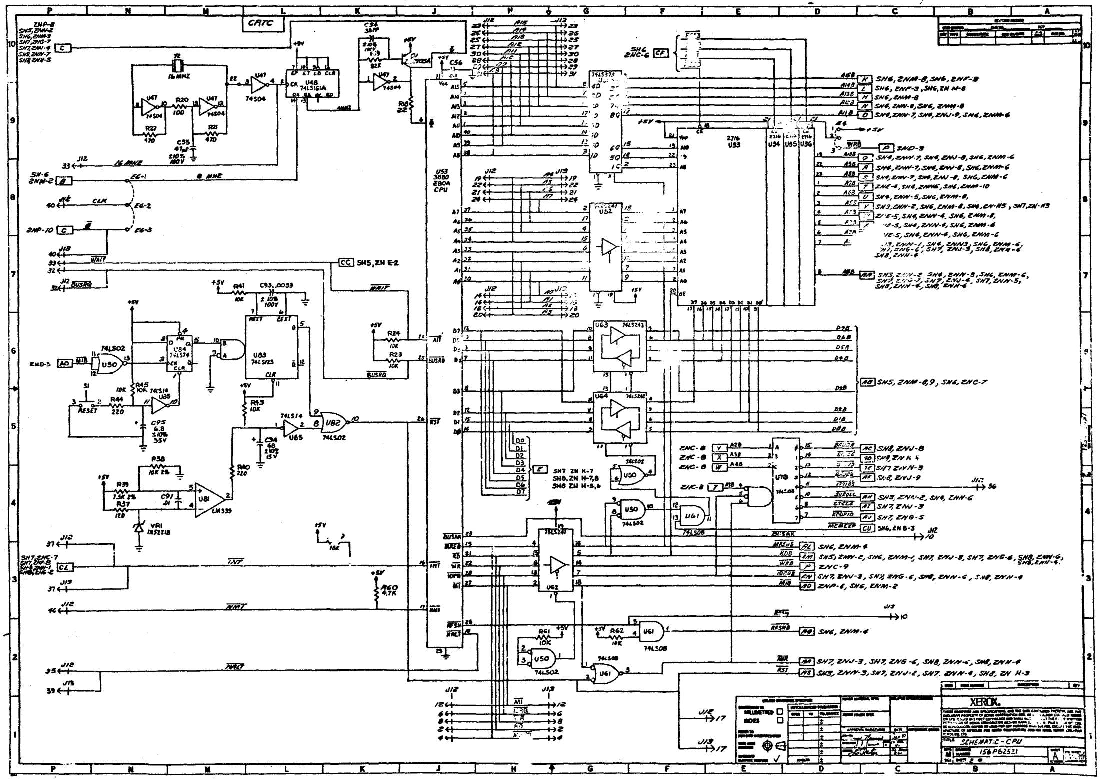


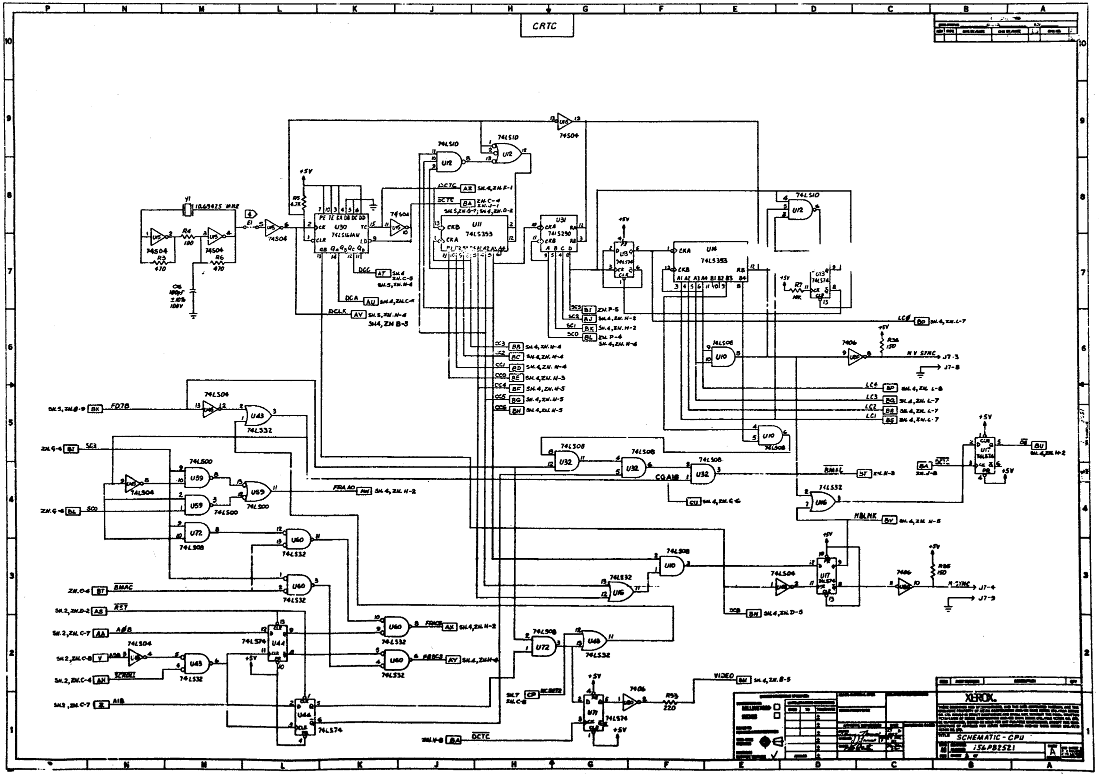
Number	Symbol	Parameter	Z-80 SIO Min	Z-80 SIO Max	Z-80A SIO Min	Z-80A SIO Max	Z-80B SIO[1] Min	Z-80B SIO[1] Max	Notes†
1	TwPh	Pulse Width (High)	200		200		200		2
2	TwPl	Pulse Width (Low)	200		200		200		2
3	TcTxC	$\overline{\text{TxC}}$ Cycle Time	400	∞	400	∞	330	∞	2
4	TwTxCl	$\overline{\text{TxC}}$ Width (Low)	180	∞	180	∞	100	∞	2
5	TwTxCh	$\overline{\text{TxC}}$ Width (High)	180	∞	180	∞	100	∞	2
6	TdTxC(TxD)	$\overline{\text{TxC}}$ ↓ to TxD Delay (x1 Mode)		400		300		220	2
7	TdTxC(W/RRf)	$\overline{\text{TxC}}$ ↓ to $\overline{\text{W/RDY}}$ ↓ Delay (Ready Mode)	5	9	5	9	5	9	3
8	TdTxC(INT)	$\overline{\text{TxC}}$ ↓ to $\overline{\text{INT}}$ ↓ Delay	5	9	5	9	5	9	3
9	TcRxC	$\overline{\text{RxC}}$ Cycle Time	400	∞	400	∞	330	∞	2
10	TwRxCl	$\overline{\text{RxC}}$ Width (Low)	180	∞	180	∞	100	∞	2
11	TwRxCh	$\overline{\text{RxC}}$ Width (High)	180	∞	180	∞	100	∞	2
12	TsRxD(RxC)	RxD to $\overline{\text{RxC}}$ ↑ Setup Time (x1 Mode)	0		0		0		2
13	ThRxD(RxC)	$\overline{\text{RxC}}$ ↑ to RxD Hold Time (x1 Mode)	140		140		100		2
14	TdRxC(W/RRf)	$\overline{\text{RxC}}$ ↑ to $\overline{\text{W/RDY}}$ ↓ Delay (Ready Mode)	10	13	10	13	10	13	3
15	TdRxC(INT)	$\overline{\text{RxC}}$ ↑ to $\overline{\text{INT}}$ ↓ Delay	10	13	10	13	10	13	3
16	TdRxC(SYNC)	$\overline{\text{RxC}}$ ↑ to $\overline{\text{SYNC}}$ ↓ Delay (Output Modes)	4	7	4	7	4	7	3
17	TsSYNC(RxC)	$\overline{\text{SYNC}}$ ↓ to $\overline{\text{RxC}}$ ↑ Setup (External Sync Modes)	-100		-100			100	2

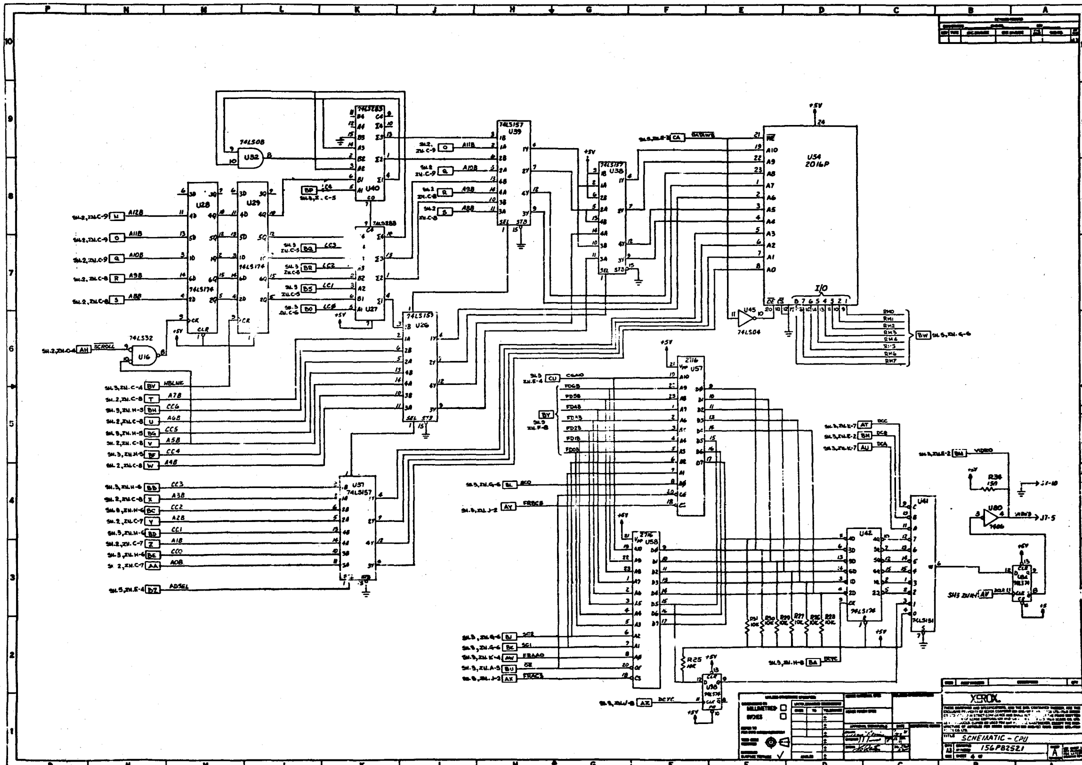
NOTES:
† In all modes, the System Clock rate must be at least five times the maximum data rate.
1. Z-80 SIO timings are preliminary and subject to change.
2. Units in nanoseconds (ns).
3. Units equal to System Clock Periods.

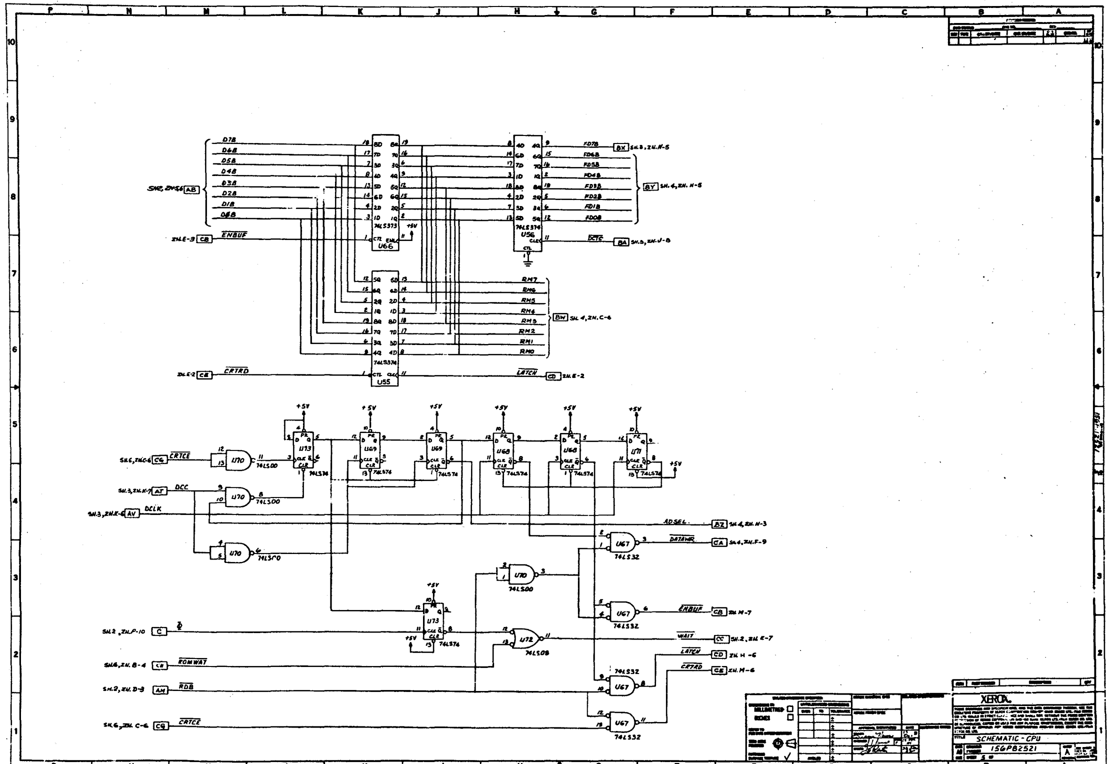
Notes

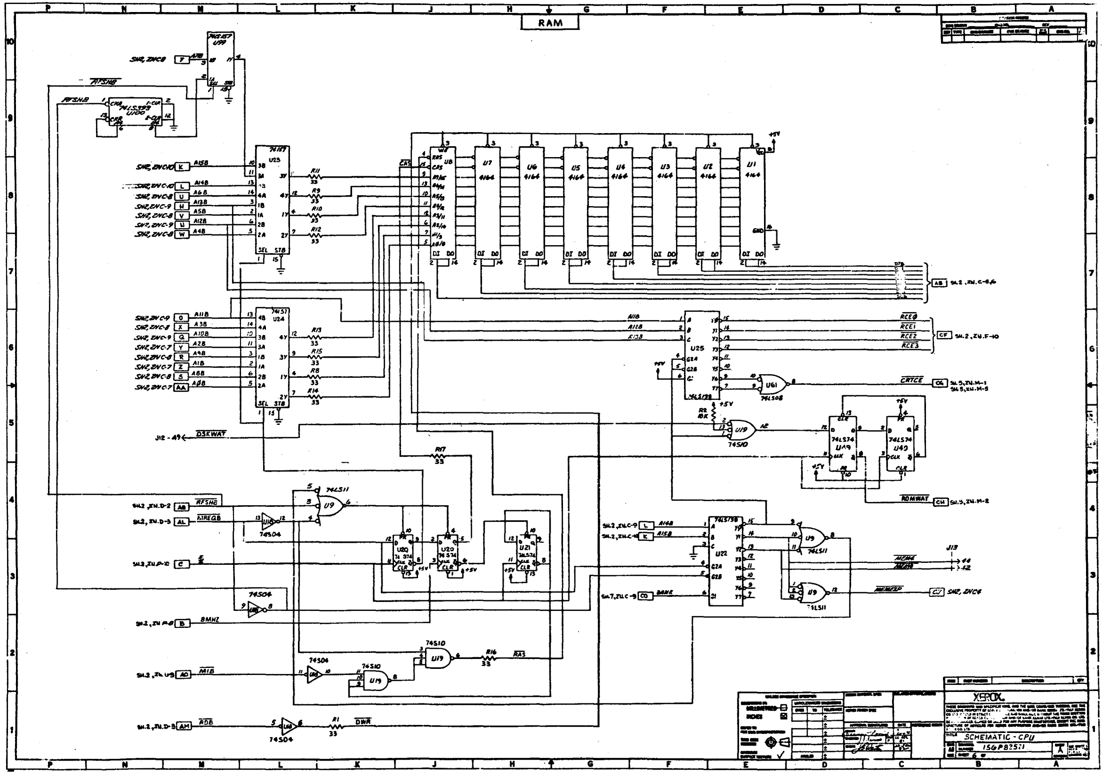

1 UNLESS OTHERWISE SPECIFIED:
RESISTANCE VALUES ARE IN OHMS, ±5%, 0.25W
CAPACITANCE VALUES ARE IN MICROFARADS,
+80 -20%, 50V
2 POWER DISTRIBUTION TABLE
REF DESIGNATIONS | GND | +5 | +12 | -12
U1-8 | 16 | 8
U9-21, 31, 32, 43, 44, 45, 47, 49, 50, 59-61, 63, 64, 67-76, 77, 79, 80, 82, 84, 85, 87, 90, 91, 93-98, 100 | 7 | 14
U81 | 12 | 3
U22-30, 37-42, 48, 78, 83, 99 | 8 | 16
U51, 52, 55, 56, 62, 66 | 10 | 20
U53 | 29 | 11
U33-36, 54, 57, 58 | 12 | 24
U75 | 31 | 9
U76 | 11 | 2 | 9
U89 | 5 | 24
U92, 65 | 11 | 26
U86, 88 | 7 | | 14 | 1
U46 | 1, 2 | 8
3 REFERENCE DESIGNATIONS
LAST USED | NOT USED
C118 | C36-39, 49, 81, 82, 108-117
E6
J13
R62
S1
U98
Q1
VR1
LS1
E7
4 FOR NORMAL OPERATIONS SHUNTS TO BE INSTALLED IN THE FOLLOWING POSITIONS
REF DESIG | BETWEEN PINS
E1 | 1, 2
E2 | 1, 2
E4 | 1, 2
E6 | 2, 3
E7 | 1, 2
J9
J10 | 3 & 4, 7 & 8
5 LAST INTERCONNECT LETTER USED (CU)
6 SPARE GATES:
74LS02
74S04
74LS04
7406
74LS08
74LS14
74LS74
74LS123
J5-8
J5-9
J5-4
J5-5
J5-6
J5-2
J13-45
J12-45
J5-1
J5-3
J5-7
+5V
GND
+12V
-12V
C107
C118
+12VB
+12VC
J12-51
J6-2
SCHEMATIC - CPU
156P82521
XEROX

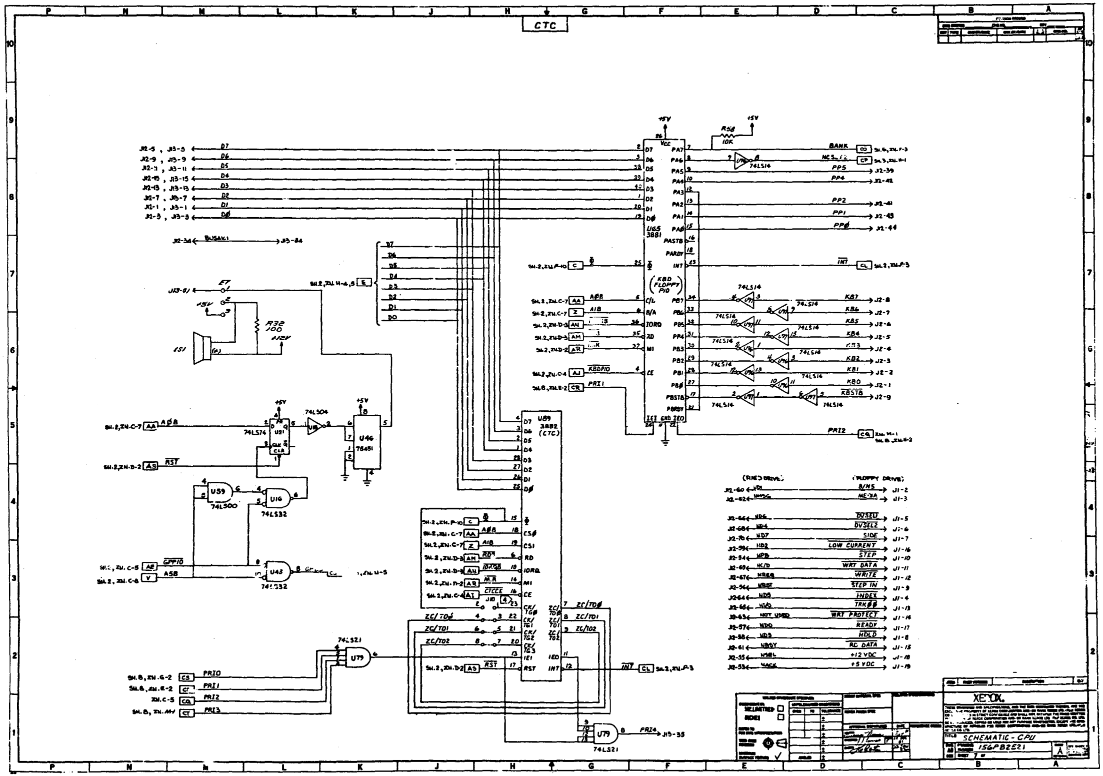

XEROX
SCHEMATIC - CPU
156P62521
CRTC

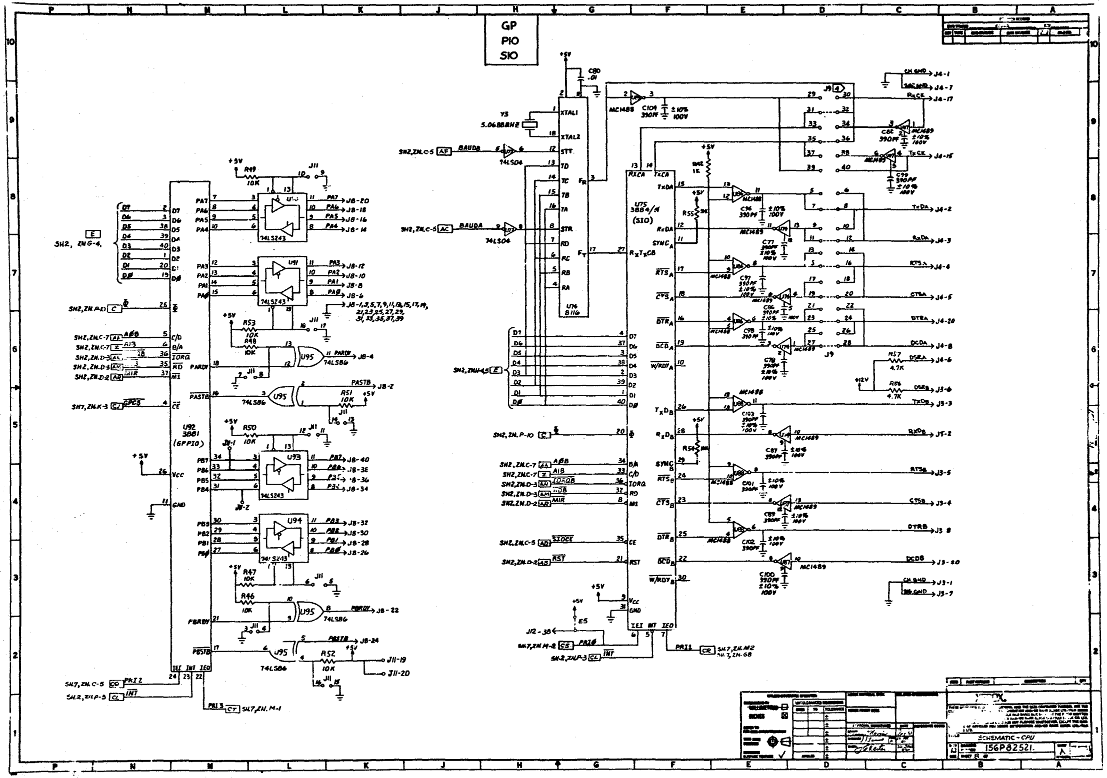

CRTC
XEROX
SCHEMATIC - CPU
156P82521

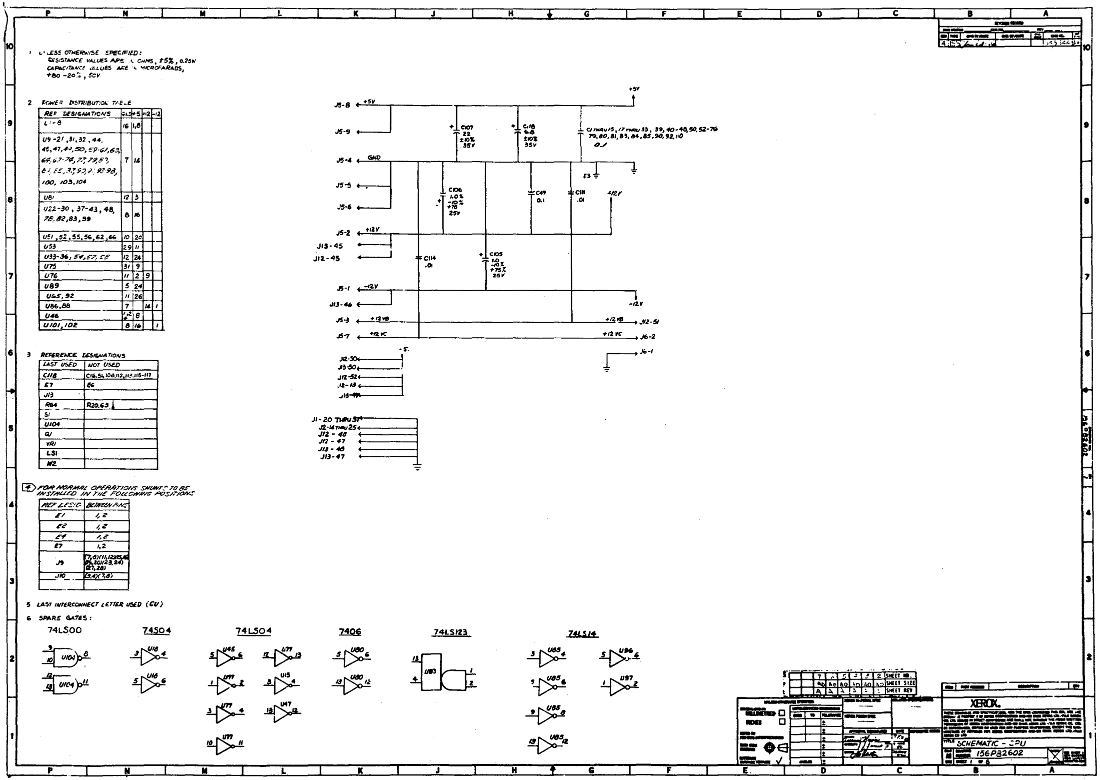

XEROX
SCHEMATIC - CPU
156PB2521
U54
2016P
U57
2116
U58
2716
U39
74LS157
U40
74LS283
U27
74LS283
U28
74LS174
U29
74LS174
U26
74LS157
U51
74LS157
U32
74LS08
U16
74LS32
U45
74LS04
U42
74LS176
U41
74LS151
U80
U38
74LS157

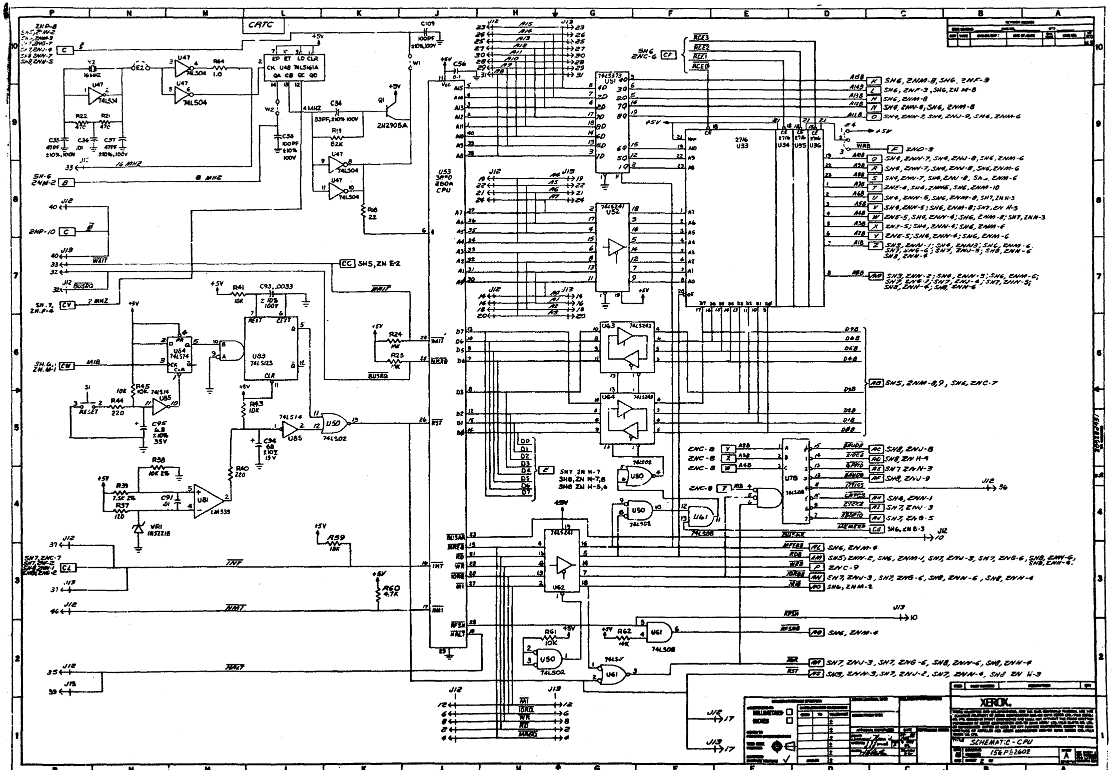

XEROX
SCHEMATIC - CPU
156PB2521
74LS373
74LS374
74LS00
74LS32
74LS08
ENBUF
CRTRD
LATCH
ADSEL
DATAWR
WAIT
ROMWAT
RDB
CRTCE
DCC
DCLK

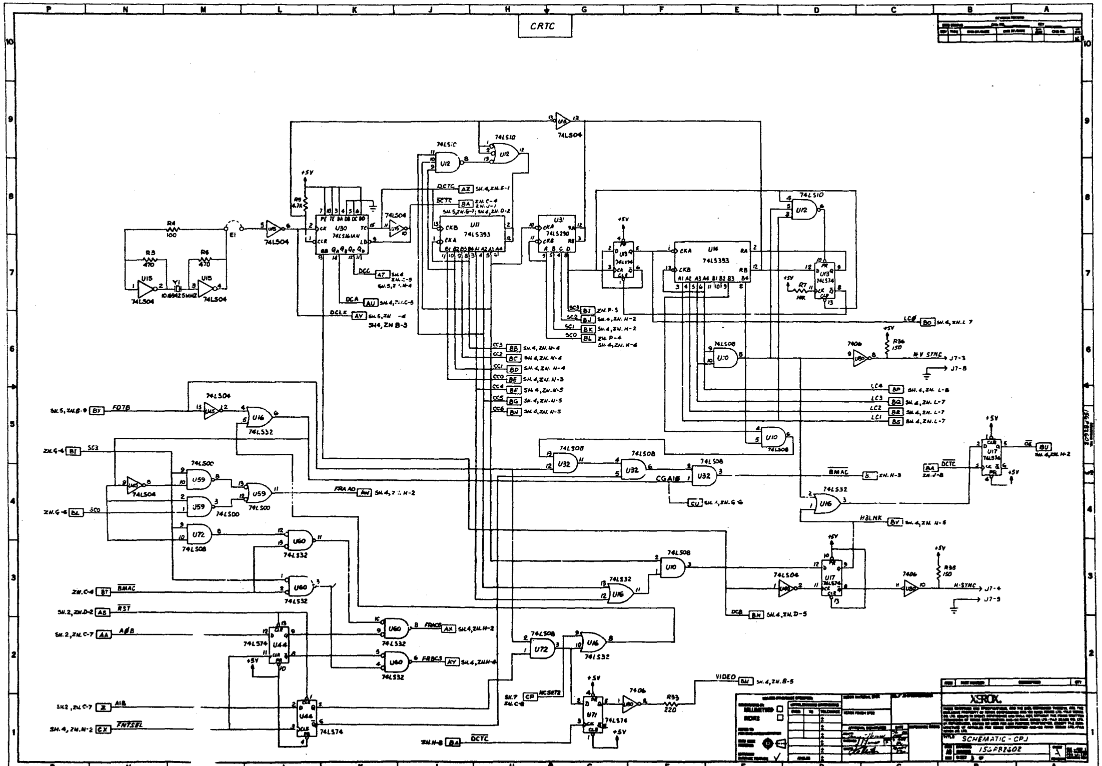

RAM
4164
74LS157
74LS393
74LS138
74LS74
74LS11
74S04
74S10
XEROX
SCHEMATIC - CPU

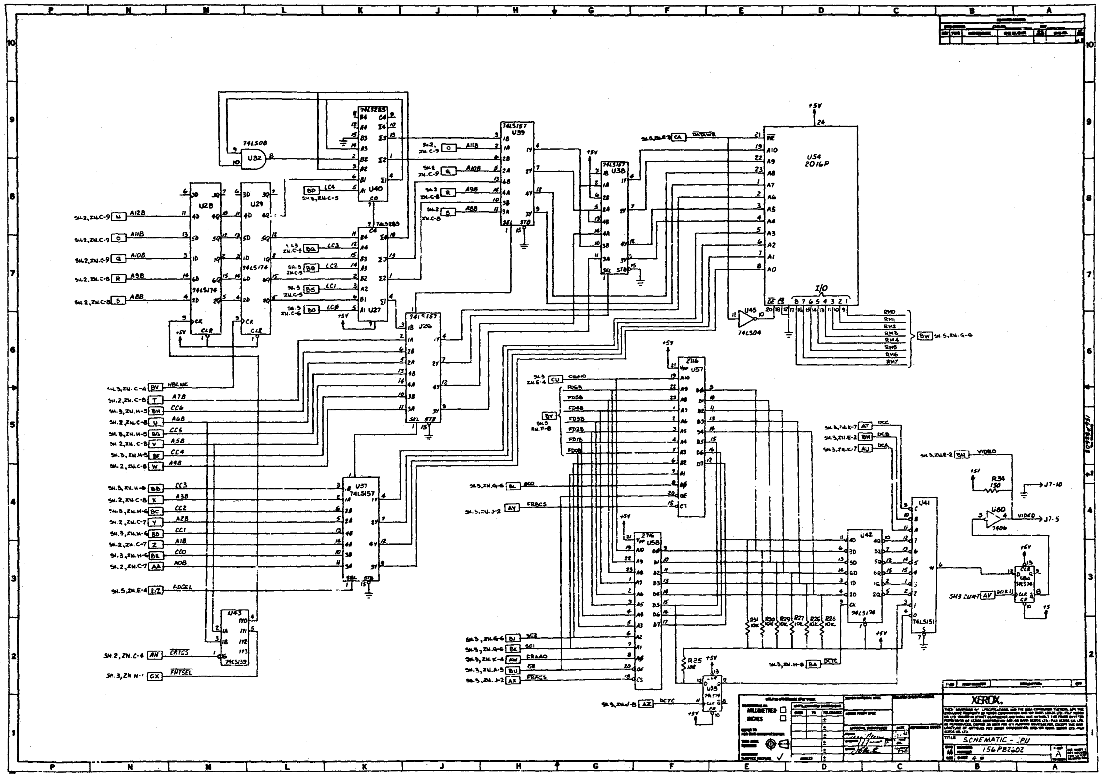

CTC
U65
3881
U89
3882
(CTC)
U46
76451
U59
74LS00
U16
74LS32
U43
74LS32
U79
74LS21
XEROX
SCHEMATIC - CPU

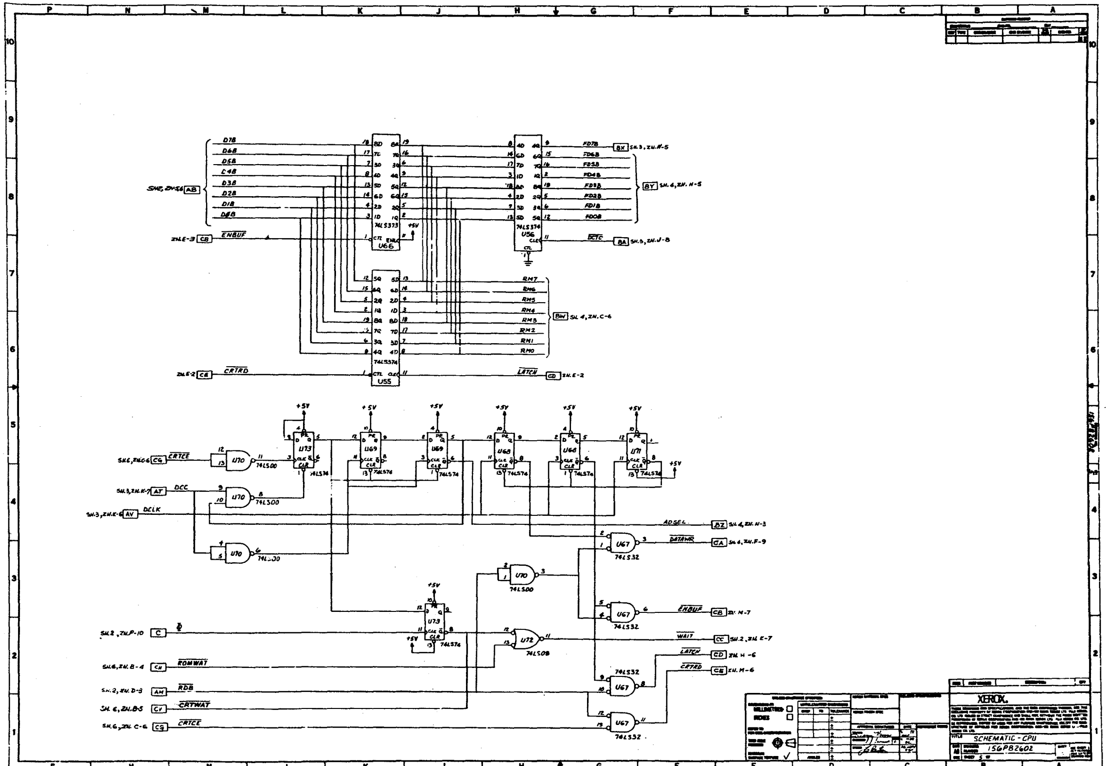

GP
PIO
SIO
U92
3881
(GPPIO)
U75
3884/4
(SIO)
U76
8116
Y3
5.0688MHZ
SCHEMATIC - CPU
156P82521

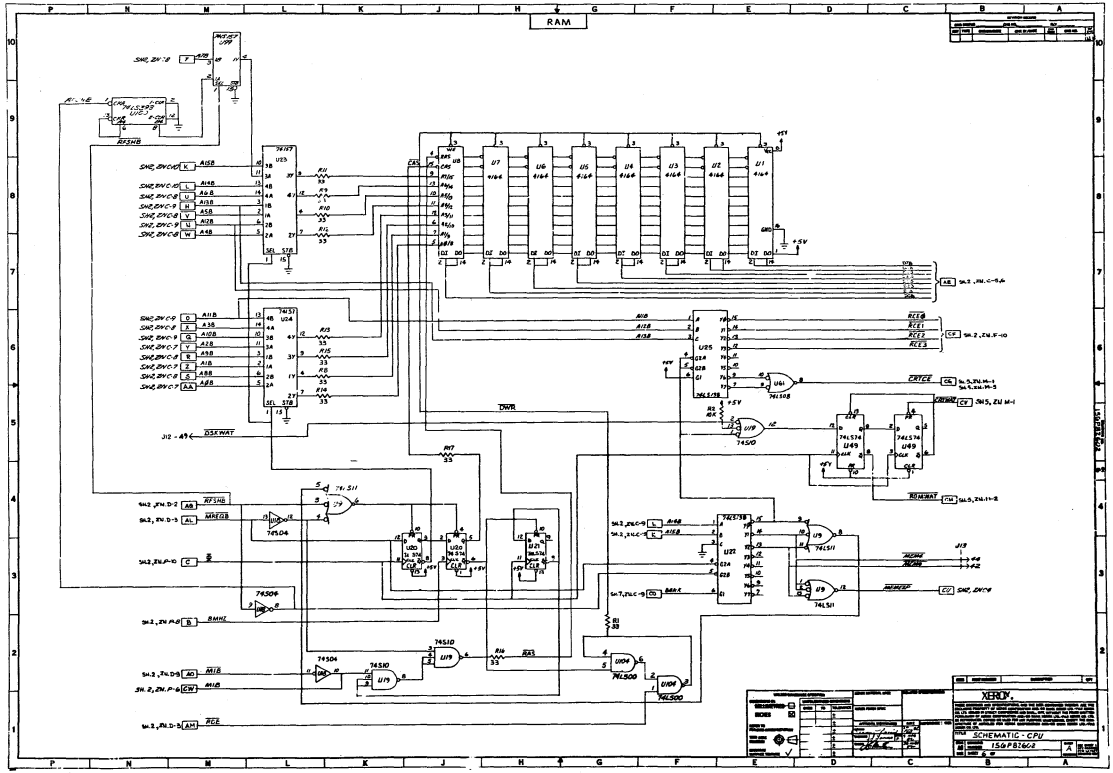

74LS00
74S04
74LS04
7406
74LS123
74LS14
XEROX
SCHEMATIC - CPU
156P82602

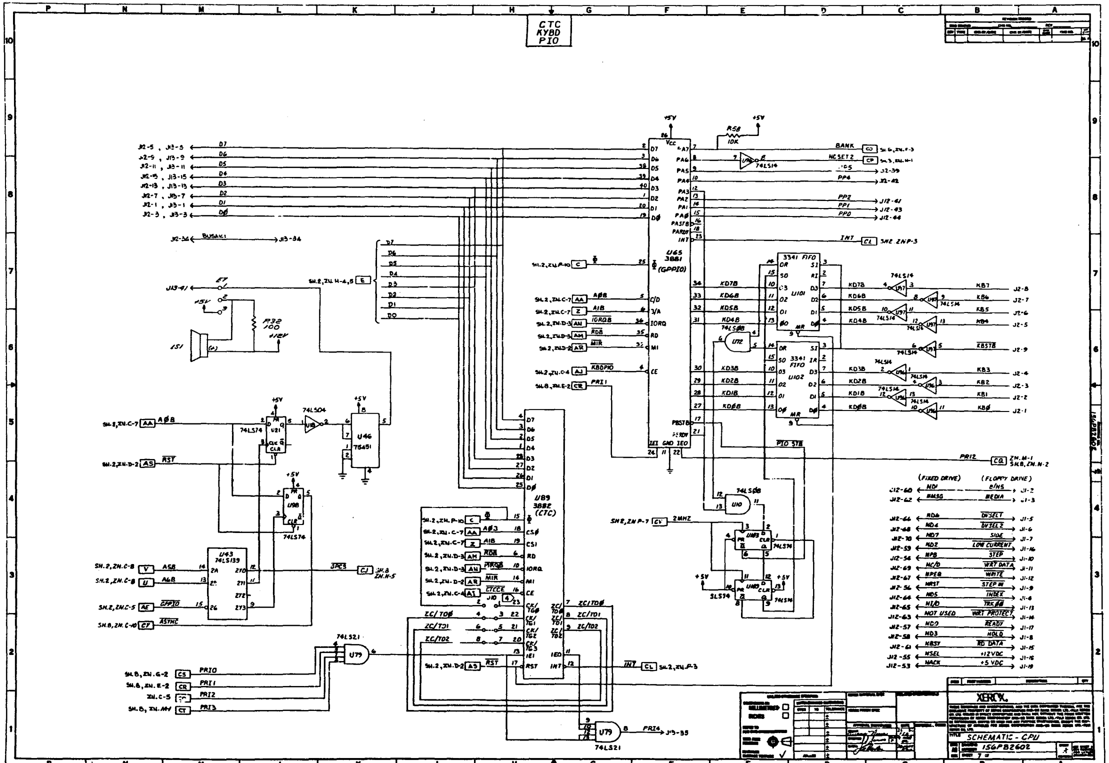

CRTC
Z80A CPU
U53
74LS373
74LS241
74LS243
2716
XEROX
SCHEMATIC-CPU
156P82602

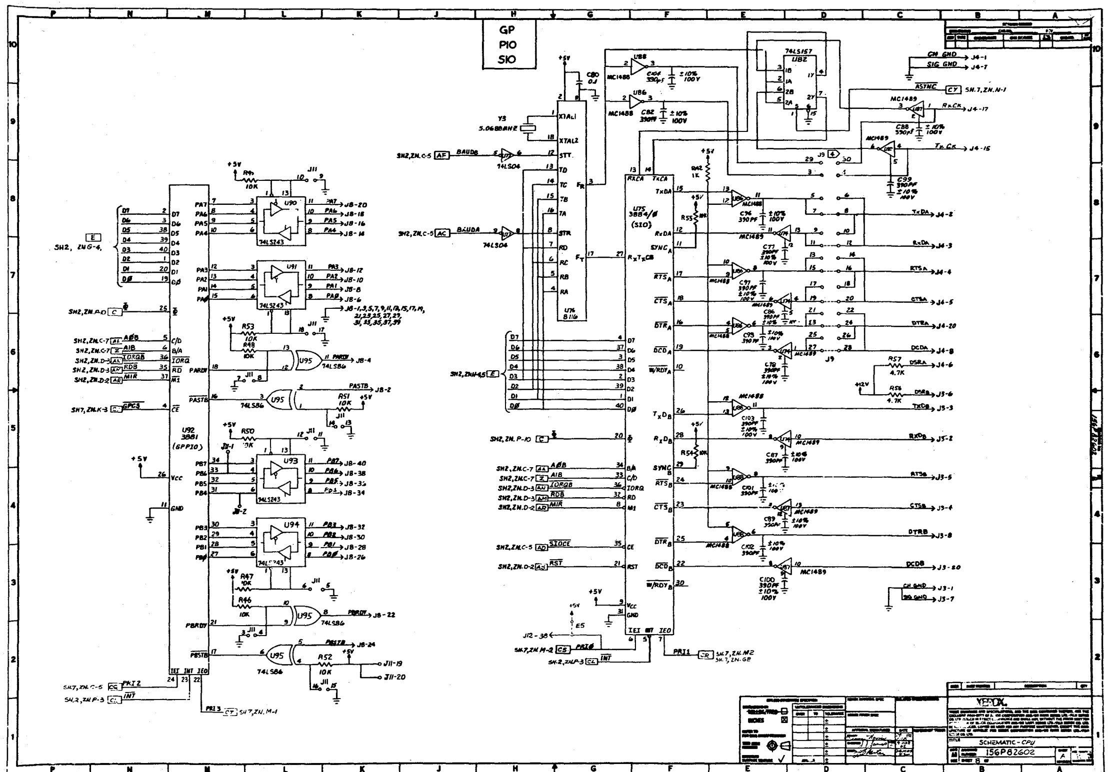

CRTC
XEROX
SCHEMATIC - CPJ
74LS04
74LS393
74LS08
74LS32
74LS74
74LS00
74LS10
H-SYNC
V SYNC
VIDEO
HBLNK


XEROX.
156PB2602
U54
2016P
74LS157
74LS174
74LS283
2716
2116
74LS04
74LS139
74LS151


D7B
D6B
D5B
D4B
D3B
D2B
D1B
D0B
74LS373
U66
ENBUF
74LS374
U56
DCTC
74LS374
U55
CRTRD
LATCH
RM7
RM6
RM5
RM4
RM3
RM2
RM1
RM0
+5V
U73
U69
U68
U71
74LS74
U70
74LS00
CRTCE
DCC
DCLK
ADSEL
DATAWR
U67
74LS32
U72
74LS08
WAIT
ROMWAT
RDB
CRTWAT
XEROX.
SCHEMATIC - CPU
156PB2602


RAM
U7 4164
U6 4164
U5 4164
U4 4164
U3 4164
U2 4164
U1 4164
U23
U24
U25
74LS138
U22
74LS04
74S10
74LS00
74LS11
74LS74
U49
DWR
DSKWAT
RFSHB
MREQB
BMHZ
RAS
CAS
CRTCE
ROMWAT
MEMWAIT
XEROX.
SCHEMATIC - CPU
156P82602


CTC
KYBD
PIO
U65
3881
(GPPIO)
U89
3882
(CTC)
3341 FIFO
U101
U102
XEROX
SCHEMATIC - CPU
15GPB2602


GP
PIO
SIO
U92
3881
(GPPIO)
U75
3884/0
(SIO)
U76
8116
74LS157
U82
74LS243
74LS86
74LS04
MC1488
MC1489
Y3
5.0688MHZ
SCHEMATIC - CPU
156P82602
XEROX

NOTES: UNLESS OTHERWISE SPECIFIED:

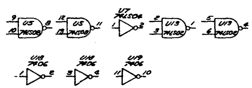
1. RESISTANCE VALUES ARE IN OHMS, ±5%, .25W
2. CAPACITANCE VALUES ARE IN MICROFARADS, +80, -20%, 50V
3. POWER DISTRIBUTION TABLE

REF DESIGNATION	GND	+5	+12
U4, U5, U7-U14, U16-U21, U23	7	14	
U1-U3, U15, U22	8	16	
U6	20	21	40

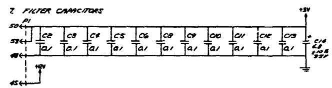
4. REFERENCE DESIGNATIONS

LAST USED	NOT USED
U23	
R11	
C14	

5. LAST INTERCONNECT LETTER USED "H"
6. SPARE GATES:



7. FILTER CAPACITORS



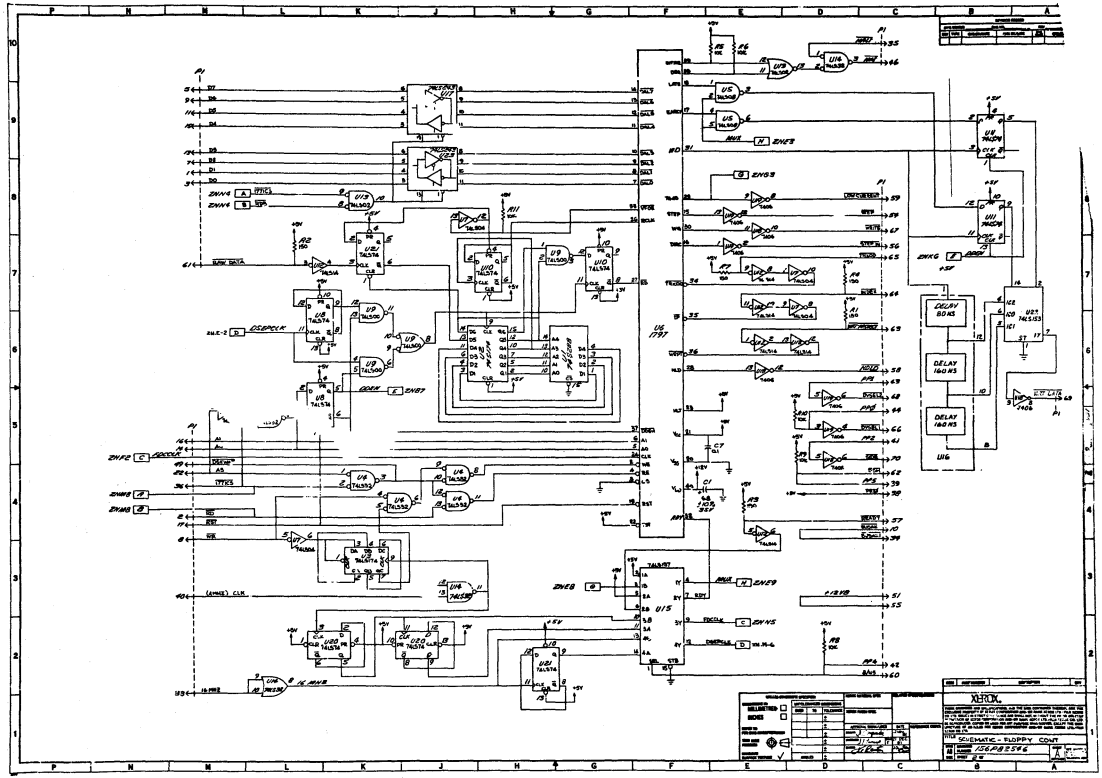
2	SHEET NO.	
AG	SHEET SIZE	
A	SHEET REV	

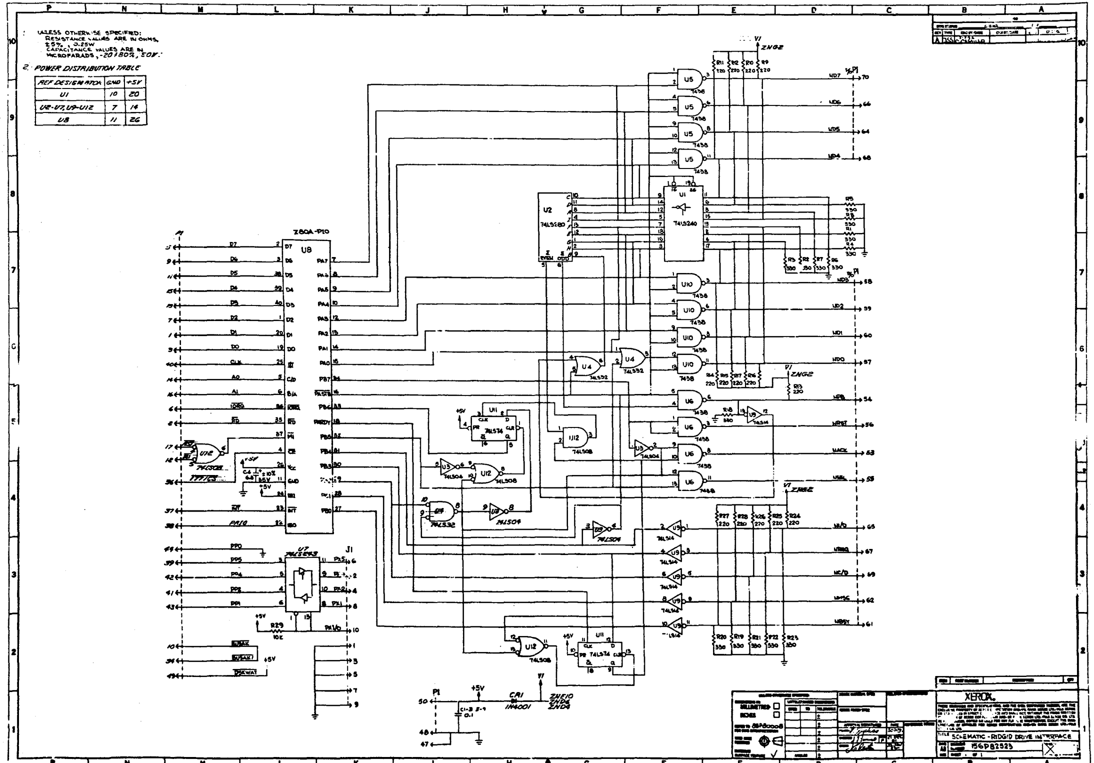
XEROX.

TITLE SCHEMATIC, FLOPPY CONTROLLER

156P82546

SHEET 1 OF 6


XEROX
SCHEMATIC - FLOPPY CONT
156P82546
U6 1797
U15
U16
U29 74LS153
DELAY 80 NS
DELAY 160 NS
DELAY 160 NS

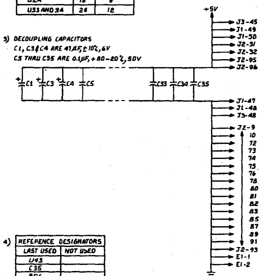

UNLESS OTHERWISE SPECIFIED:
RESISTANCE VALUES ARE IN OHMS, ±5%, 0.25W
CAPACITANCE VALUES ARE IN MICROFARADS, -20/80%, 50V.
2. POWER DISTRIBUTION TABLE
REF DESIGNATOR GND +5V
U1 10 20
U2-U7, U9-U12 7 14
U8 11 26
Z80A-PIO
U8
74LS240
74LS280
74LS74
74LS08
74LS04
74LS32
7438
74LS14
XEROX
SCHEMATIC - RIDGID DRIVE INTERFACE
156P82523

1) UNLESS OTHERWISE SPECIFIED:
RESISTANCE VALUES ARE IN OHMS, ± 5%, .25W
CAPACITANCE VALUES ARE IN MICROFARADS + 100, -0%, 50 WVDC

2) POWER DISTRIBUTION TABLE

REF DESIGNATION	PIN DESIGNATIONS	
	+5V	GND
U1 THRU U16	8	16
U17, 19, 20, 21, 23, 25 THRU 32 AND 36 THRU 42	20	10
U18 AND 35, 43	14	7
U22	40	1 & 20
U24	18	9
U33 AND 34	24	12

3) DECOUPLING CAPACITORS
C1, C3 & C4 ARE 47µF, ± 10%, 6V
C5 THRU C35 ARE 0.1µF, + 80 - 20%, 50V



4)

REFERENCE DESIGNATORS	
LAST USED	NOT USED
U43	
C35	
RP6	
J3	
Y1	

J1 CONNECTOR

PIN NO.	SIGNAL NAME	SHEET/ZONE
1	Z8DØ1	SH3, C7
2	NZ8RD	SH3, N3
3	Z8DØØ	SH3, C7
4	N.C.	
5	Z8DØ7	SH3, C7
6	NZ8IRQ	N8
7	Z8DØ2	C7
8	NZ8WR	N3
9	Z8DØ6	SH3, C7
10	N.C.	
11	Z8DØ5	SH3, C7
12	NZ8M1	SH3, N3
13	Z8DØ3	SH3, C7
14	Z8ADØØ	SH2, N6
15	Z8DØ4	SH3, C7
16	Z8ADØ1	SH3, N9
17	N.C.	
18	Z8ADØ2	SH3, N9
19	Z8ADØ4	N9
20	Z8ADØ3	N9
21	Z8ADØ6	N7
22	Z8ADØ5	SH3, N8
23	N.C.	
24	Z8ADØ7	SH3, N7
25	N.C.	
26	N.C.	
27	N.C.	
28	Z8AD10	SH3, N6
29	Z8ADØ9	SH3, N6
30	Z8AD11	SH3, N3
31	Z8ADØ8	SH3, N7
32	N.C.	
33	NZ8WAIT	SH3, J5
34	N.C.	
35	N.C.	
36		
37	N.C.	
38		
39	N.C.	
40	N.C.	
41	N.C.	
42	NZ8MEMB	SH3, N6
43		
44	NZ8MEM4	SH3, N6
45	12 VOLTS	SH2, C2
46	N.C.	
47	GND	
48	GND	
49	+5V	
50	+5V	

J2 CONNECTOR

PIN NO.	SIGNAL NAME	SHEET/ZONE	PIN NO.	SIGNAL NAME	SHEET/ZONE
1	EXADØØ	SH4, B8	49		
2	EXADØ1	B8	50		
3	EXADØ2	B8	51		
4	EXADØ3	B7	52		
5	EXADØ4	B7	53		
6	EXADØ5	B7	54		
7	EXADØ6	B7	55		
8	EXADØ7	SH4, B7	56		
9	GND		57		
10	GND		58		
11	NAMWC	SH2, F1	59		
12	DRADØ	SH4, G2	60		
13	DRAD1		61		
14	DRAD2		62		
15	DRAD3		63		
16	DRAD4		64		
17	DRAD5		65	12 VOLTS	SH2, C2
18	DRAD6		66	ADØØ	SH2, C10
19	DRAD7	G2	67	NZLOCK	SH3, N6
20	NCAS2	G1	68		
21	EXADØ8	B5	69	ALE	SH2, F3
22	EXADØ9	B5	70		
23	EXAD10	B4	71		
24	EXAD11	B4	72	GND	
25	EXAD12	B4	73	GND	
26	EXAD13	B4	74	GND	
27	EXAD14	B3	75	GND	
28	EXAD15	B3	76	GND	
29	NCAS3	B1	77	NEXPCAS	SH4, K2
30	NRAS	SH4, B1	78	GND	
31	+5V		79	N8OLOCKED	SH3, N6
32	+5V		80	GND	
33			81	GND	
34			82	GND	
35			83	GND	
36			84	NINTA	SH2, F1
37			85	GND	
38			86	NBHE	SH2, C4
39			87	GND	
40			88	AD19	SH2, C5
41			89	GND	
42			90	AD18	SH2, C4
43			91	GND	
44			92	AD17	SH2, C4
45			93	GND	
46			94	NZINTREQ	SH2, C2
47			95	+5V	
48			96	+5V	

J3 CONNECTOR

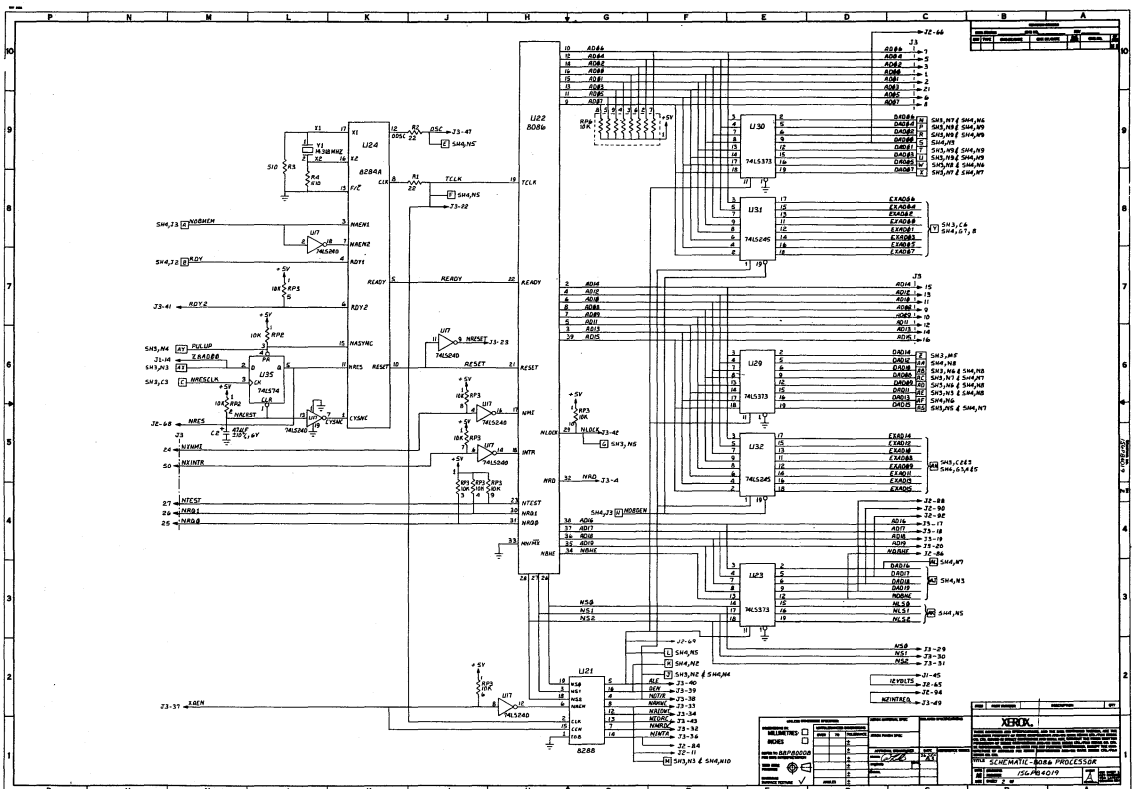
PIN NO.	SIGNAL NAME	SHEET/ZONE
1	ADØØ	SH2, C10
2	ADØ1	C10
3	ADØ2	C10
4	NRD	G5
5	ADØ4	C10
6	ADØ5	C9
7	ADØ6	C10
8	ADØ7	C9
9	ADØ8	C7
10	ADØ9	C7
11	AD10	C7
12	AD11	C6
13	AD12	C7
14	AD13	C6
15	AD14	C9
16	AD15	C6
17	AD16	C4
18	AD17	C4
19	AD18	C4
20	AD19	C4
21	ADØ3	C9
22	TCLK	J8
23	NRESET	H6
24	NINMI	N5
25	NRØØ	N4
26	NRØ1	N4
27	NTEST	SH2, N4
28	NRAS	SH4, G1
29	NSØ	SH2, C2
30	NS1	C2
31	NS2	C2
32	NMRDC	F1
33	NAMWC	F2
34	NAIOWC	SH2, F1
35	NEXPCAS	SH4, K2
36	NINTA	SH2, F1
37	NAEN	H2
38	NDT/R	F2
39	DEN	F2
40	ALE	F2
41	RDY2	N7
42	NLOCK	G5
43	NIORC	SH2, F1
44	NASEL	SH4, J5
45	+5V	
46	NLBUSCY	SH4, K2
47	OSC	SH2, J3
48	GND	
49	NZINTREQ	SH2, C2
50	NZINTR	SH2, N5

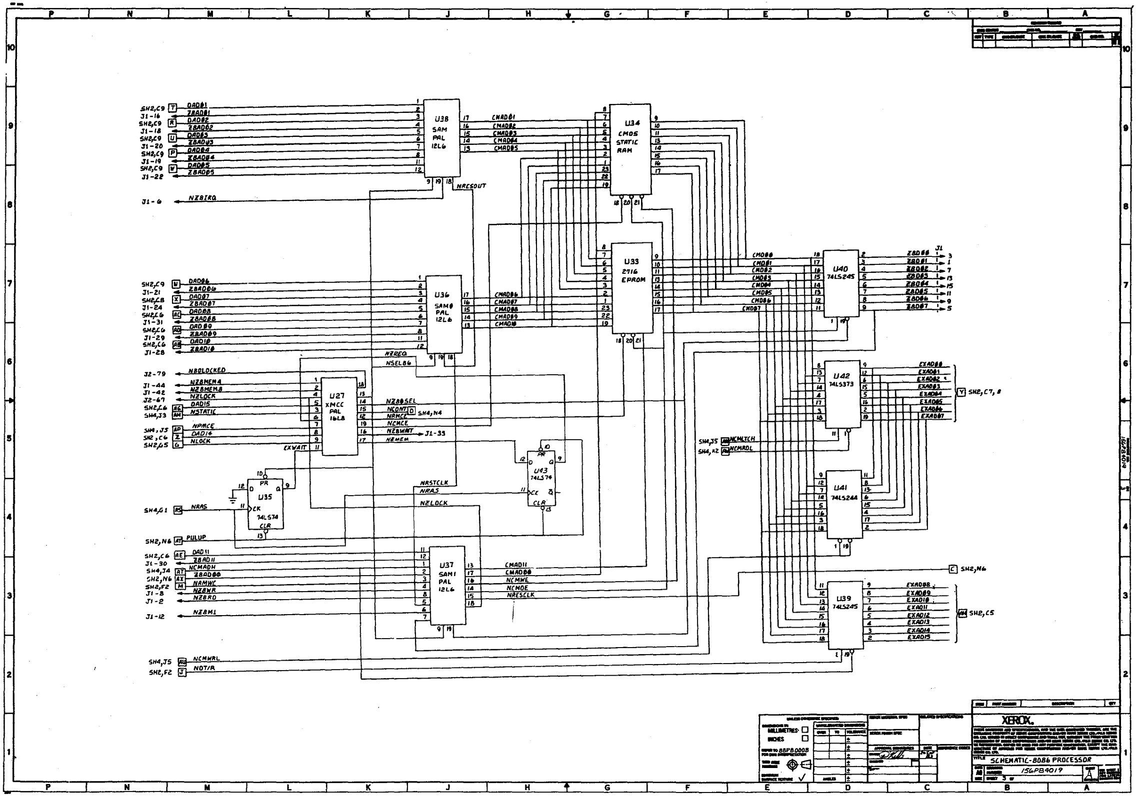
XEROX

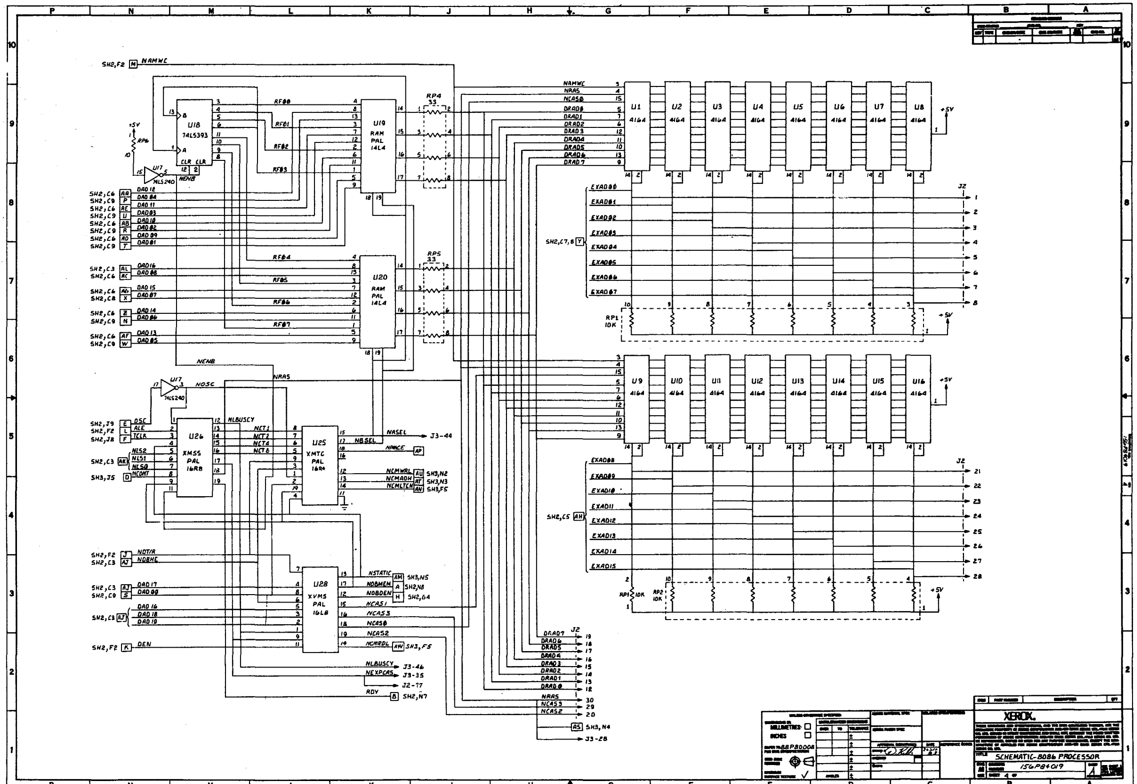
SCHEMATIC - 8086 PROCESSOR

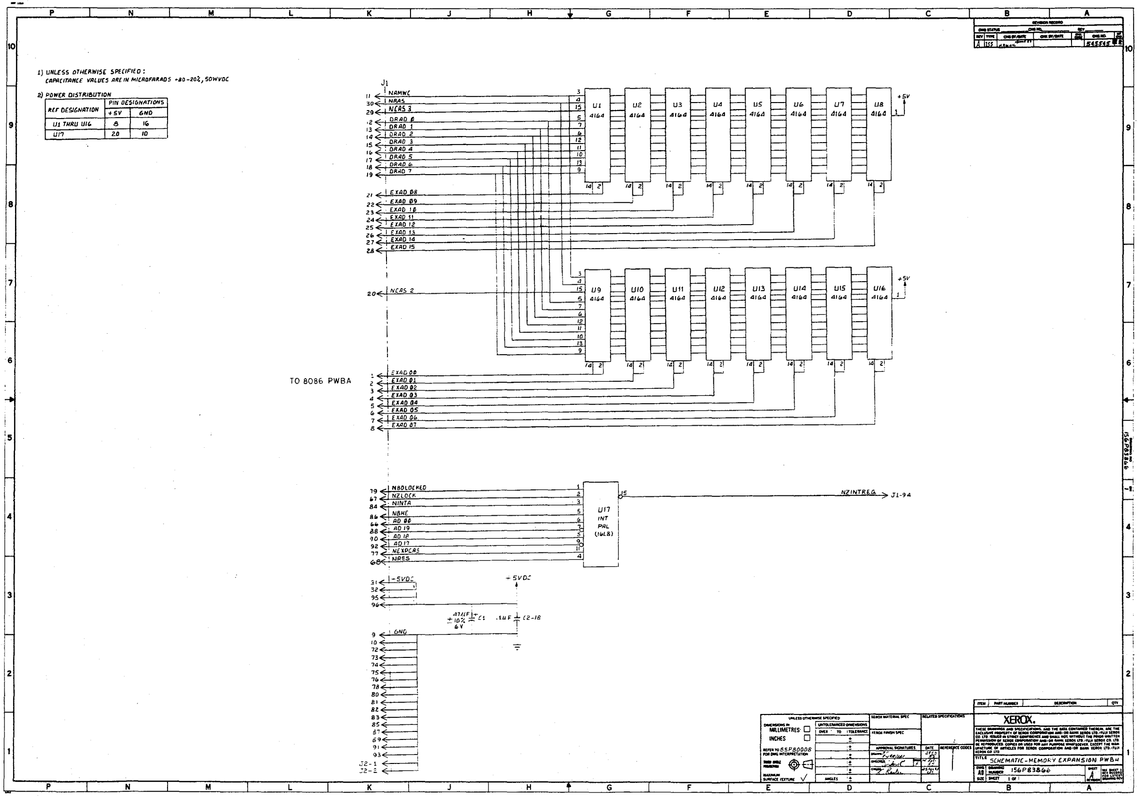
156P84019

SHEET 1 OF 4


XEROX
SCHEMATIC-8086 PROCESSOR
15GP84019
U22
8086
U24
8284A
U21
8288
U30
74LS373
U31
74LS245
U29
74LS373
U32
74LS245
U23
74LS373
U35
74LS74
U17
74LS240


U38 SAM PAL 12L6
U34 CMOS STATIC RAM
U33 2716 EPROM
U40 74LS245
U36 SAM0 PAL 12L6
U27 XMCC PAL 16L8
U42 74LS373
U43 74LS74
U35 74LS74
U41 74LS244
U37 SAM1 PAL 12L6
U39 74LS245
XEROX
SCHEMATIC-8086 PROCESSOR
156P84019


U18
74LS393
U19
RAM
PAL
14L4
U20
RAM
PAL
14L4
U26
XMSS
PAL
16R8
U25
XMTC
PAL
16R4
U28
XVMS
PAL
16L8
4164
XEROX
SCHEMATIC-8086 PROCESSOR
156P84019


1) UNLESS OTHERWISE SPECIFIED:
CAPACITANCE VALUES ARE IN MICROFARADS +80-20%, 50WVDC
2) POWER DISTRIBUTION
REF DESIGNATION | PIN DESIGNATIONS +5V | GND
U1 THRU U16 | 8 | 16
U17 | 20 | 10
J1
TO 8086 PWBA
NAMWC
NRAS
NCAS 3
NCAS 2
DRAD 0 … DRAD 7
EXAD 00 … EXAD 15
U1 … U16 4164
+5V
NBOLOCKED
NZLOCK
NINTA
NBHE
AD 00
AD 19
AD 18
AD 17
NEXPCAS
NRES
U17 INT PAL (16L8)
NZINTREQ
J1-94
+5VDC
GND
47UF ±10% 6V C1
.1UF C2-18
XEROX.
SCHEMATIC-MEMORY EXPANSION PWBA
156P83866

How the ROM Works

When power is first applied or the RESET button is pressed, the system bank (Bank 0) is enabled by hardware and the Z80-A's program counter is set to 0000H.

The first 8k of address space in the system bank consists of ROM sockets U33-U36. An 820-II with the ASCII keyboard will have 6k of ROM occupying U33-U35 (6k). With the Low Profile keyboard, another 2k ROM is added to U36 bringing the total system ROM size to 8k.

The firmware contained in the system ROMs will be referred to as the ROSR (ROM Operating System Routines). The ROSR provides instructions for the Z80-A to do several things at power-on. They are:

> Do a checksum test of the firmware contained in the first 6k of ROM.
>
> Do confidence test on RAM memory that will be used by the ROSR (F000H-FFFFH).
>
> Initialize programmable devices and variable memory area starting at address FF00H.
>
> Move resident portion of monitor to RAM starting at address F000H.
>
> Compute checksum of the ROM in socket U36; if correct, call the first address of U36 (1800H).
>
> Check type of disk controller daughter board that is installed and load appropriate disk driver into high memory.
>
> Provide an initial system command level for the user. This provides such options as: Host terminal mode, Typewriter mode, Load system, Dump memory, etc.

Additionally, ROSR provides character I/O, disk I/O, and other hardware-related services for the operating system and /or an application program.

One of the first things that the ROSR is responsible for is the initialization of the programmable devices on the CPU board. These devices are: CTC

(Counter Timer Circuit), PIOs (Parallel Input Output controllers), SIO (Serial Input Output controller), and the Baud rate generator. At power on, these devices are initialized as described below.

Counter Timer Circuit (CTC)

Base interrupt vector = 0H

Channel 0	Not initialized
Channel 1	Timer mode, no interrupts, period = 1 msec.
Channel 2	Timer mode, no interrupts, period = 8 msec.
Channel 3	Counter mode, interrupts enabled, down counter value = 125.

Channel 0 of the CTC is not used by the system. Channel 1 is initialized but interrupts are not enabled until the screen print command is given from the keyboard. At this time, 1 msec. interrupts begin occurring until the last character has been printed from the CRT's refresh memory. Then, the interrupts from CTC-1 will be disabled. Channel 2 is initialized as a timer also. Its job is to divide the system clock and to generate a pulse to CTC-3 every 8 msec. Channel 3 is initialized as a counter; it counts pulses from CTC-2 and generates an interrupt to the system every 125 pulses (1 second).

System Parallel Input/Output Controller (PIO)

Base interrupt vector = 1AH

Port A	Bit Mode	Bits 0-5 input, Bits 6-7 output, interrupts disabled
Port B	Input Mode	Interrupts enabled

Port A of the system PIO is used for the bank switching, floppy disk drive and side selects, and CRT font selection. Port B is used as the keyboard input channel.

General Purpose Parallel Input/Output Controller (PIO)

Base interrupt vector	=	Not initialized
Port A Output Mode		Interrupts disabled
Port B Mode 3		Bits 0-3 output, bits 4-7 input, interrupts disabled

The general purpose parallel PIO is initialized to provide an interface to a Centronics-compatible parallel printer. Port A serves as the data channel and port B bit 2 provides the strobe to the printer. Port B bit 4 is for connection to the printer's ready signal. The parallel interface option connector (J11) must have jumpers installed between the following pins: 5-6, 9-10, and 17-18. This selects the direction for the transceiver; that is, between the PIO and the parallel I/O connector J8.

If the GP PIO is to be used for something other than a parallel printer, the user can re-program the PIO and re-jumper J11 to suit the needs of the application.

Serial Input/Output Controller (SIO)

Base interrupt vector = 00

Channel A — Asynchronous mode modem port
Interrupts disabled, 7 bits per character, x16 clock mode, 1 stop bit per character, even parity enabled, Data Terminal Ready (DTR) and Request To Send (RTS) outputs from the SIO are active. Wired as RS-232 DTE (Data Terminal Equipment).

Note: The 4.03 ROM initializes DTR and RTS outputs to an inactive state.

Channel B — Printer port
Interrupts disabled, 7 bits per character, x16 clock mode, 1 stop bit per character, even parity enabled, Data Terminal Ready (DTR)

and Request To Send (RTS) outputs from the SIO are active. Wired as RS-232 DCE (Data Communication Equipment). Hardware handshake is available on pins 20 and 5.

Channel A Baud Rate Generator (Modem)

300 Baud

Channel B Baud Rate Generator (Printer)

1200 Baud

MONITOR COMMANDS

The 820-II and 16/8's resident monitor is capable of executing several commands directly from the keyboard.

The table below summarizes the monitor's command set. Under the heading "Format", the items enclosed in **parentheses** represent **required** numeric parameters. The items in **square brackets** represent **optional** parameters. Detailed information on each of the commands follows the table.

Command	Format
D(isplay memory)	D [start addr] [end addr]
M(odify memory)	M (addr)
X(tended memory test)	X (start addr) (end addr)
F(ill memory)	F (start addr)(end addr)(fill data)
C(opy memory)	C (start addr)(end addr)(dest addr)
V(erify memory block)	V (start addr)(end addr)(with addr)
G(oto)	G (addr)[HL, DE, BC registers]
I(nput)	I (16-bit port addr)
O(utput)	O (16-bit port addr)(8-bit data)
L(oad from disk)	L [disk drive unit value]
R(ead disk sector)	R (drive unit)(track)(sector)(addr)
W(rite disk sector)	W (drive unit) (track) (sector) (addr)
B(aud rate)	B (baud rate)[channel]
T(ypewriter)	T [baud rate]
H(ost terminal)	H [ch.] [baud] [data bits] [par.] [stop bits]
P(rinter protocol)	P (Xon/Xoff)[status mask] [status value]

1) D - DISPLAY MEMORY COMMAND

This command displays the contents of memory in hexadecimal and ASCII representation. Each display line has the following format:

AAAA DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD CCCCCCCCCCCCCCCC

AAAA is the starting memory address of the line in hexadecimal, the DD's are the hex values of the 16 bytes of data starting at location AAAA, and the C's are ASCII characters equivalent to each data byte. Bytes with a value less than 20 hex are displayed with their appropriate display character codes as shown in the ASCII code chart. Bytes with a value greater than 7F hex are displayed in low intensity. The display memory command accepts one, two, or no address parameters. If two addresses are specified, the block of memory between those two locations will be displayed. Entering only one address will display 256 bytes of memory starting at the specified location. Entering **D<return>** with no parameters will display the 256 byte block of memory starting one location past the last address displayed.

The display can be stopped temporarily by touching the space bar. Touching the space bar again continues the display.

2) M - MODIFY MEMORY COMMAND

The modify memory command allows the contents of individual memory locations to be changed. This command accepts one parameter representing the first memory address to modify or examine. The display format is:

AAAA DD

AAAA is the current memory address and DD is the hexadecimal value of the data in that location. After displaying the contents of a memory location, the routine waits for one of the following parameters to be entered from the keyboard:

- Touching <return> does not modify the memory data at the currently displayed memory address, but will display the contents of the next memory address.

- Typing a minus sign has a similar effect, except the address is decremented instead of incremented.

- Typing a hexadecimal number will replace the data at the currently displayed address with the number entered. The new data is stored as soon as the second digit is entered, with no terminating character required. If only one digit is entered, touching <return> will cause the single digit hex number to replace the previous data.

- Typing a quote sign will cause the ASCII value of the next key typed to be stored at the currently displayed address.

- Typing any character other than <return>, a minus sign, a quote sign, or a hexadecimal digit will terminate the command.

3) X - EXTENDED MEMORY TEST COMMAND

This command tests the specified range of memory for errors. Any portion of memory may be tested except the read/write area reserved for ROSR (F000 to FFFF hex). At least two parameters are required: the starting address and the ending address.

Only the high-order eight bits of the addresses entered are actually used. If no errors are detected, the test will display a plus sign. When errors are detected an error line will be displayed in the following format:

```
AAAADD    should = X
```

4) F - FILL MEMORY COMMAND

The fill command allows blocks of memory to be filled with a fixed data value. Three parameters are required: a starting address, an ending address, and a fill-data value. Each location in the specified block of memory has the constant written into it and then read back again to check for memory errors. An error line (like the one described for the Xtended Memory Test) is printed for any locations that fail to verify.

5) C - COPY MEMORY COMMAND

The copy command allows blocks of data to be moved in memory. Three parameters are required: a starting memory address, an ending address, and a destination address. The contents of the block of memory in between the first two addresses are copied to the block starting at the third address. Like the Fill Memory command, a test is made to verify that each byte of the destination block, when read back, is the same as the corresponding byte in the source block.

6) V - VERIFY MEMORY BLOCK COMMAND

This command is similar to the Copy Memory command except that data is not moved, but simply checked to see if it is the same as data located at a different address in memory. Three parameters are required: a starting memory address, an ending address, and the first address of the memory block to be compared.

7) G - GO TO COMMAND

The go to command controls the CPU to start executing at a particular memory location. It requires a single parameter, (the address at which to begin execution). Three optional parameters may also be specified so the HL, DE, and BC register pairs may be preset before execution begins. Each of these optional parameters is a 16-bit (four-digit hexadecimal) number. The optional parameters would be entered as:

```
G1000 AAFF DDEE BBCC
```

1000 is the hexadecimal address at which to begin execution, AA is placed in the H register, FF is placed in the L register, DD is placed in the D register, EE is loaded into the E register, BB is transferred to the B register, and CC is put into the C register. In addition, FF (the value specified for the L register) will also be placed into the A register. Thus, a shortened command line (using only a single optional parameter) would be entered as:

```
GF00C 0024
```

In the example, the hex value 24 will be loaded into both the L and A registers before executing the CRT output driver at memory address F00C hex. (This example displays a '$' on the screen.)

ROSR actually passes control to the specified address by simulating a CALL instruction. This makes it possible for the external program to return to ROSR by doing a RET, (assuming it does not re-load the stack pointer or lose the return address to ROSR). After the routine returns, ROSR displays the contents of the A register and the HL register pair.

8) I - INPUT COMMAND

This command allows data to be read from input ports. It works very much like the Modify Memory command, except input ports are being examined instead of memory locations. A single parameter representing a port number is required. Since many of the I/O ports are accessed using the unique Z80-A I/O instruction, the parameter can be a 16-bit port address. The BC register pair is loaded with the parameter, and then an IN A,(C) instruction is executed. An example of full parameter specification would be:

IAA55

AA represents the contents of the B register, which is placed on the high-order address lines (A8-A15), and 55 represents the contents of the C register, which is placed on the low-order address lines (A0-A7). Using only an 8-bit parameter will place a zero in the B register.

Touching the space bar will display data from the same port address again. The contents of adjacent ports can be examined by touching <return> or the minus sign (like the Modify command). Typing any other key terminates the command.

9) O - OUTPUT COMMAND

The output command allows a specified data value to be written to output ports. Two parameters are required: a 16-bit port address (see Input command), and an 8-bit data value that is to be written to that port. After outputting the specified data to the port, the command returns to ROSR instead of stepping to the next output port like the input command. This makes it possible to use the output command to initialize Z80-A peripheral devices like the SIO, PIO, and CTC. Since a 16-bit port address is specified, special ports such as the scroll port register can be modified directly from ROSR. Some of these special ports require that their "data" value be placed on the high-order address lines. A sample command to alter the scroll port register is:

O1014FF

10 represents the contents of the B register which is placed on the high-order address lines and is the actual "data" that will be written to the scroll port register. The 14 represents the value that is placed in the C register and is output as the low-order address lines to actually select the scroll port. The data value FF hex is output on the data lines, but the data is not looked at by this type of special port.

10) L - LOAD FROM DISK

The load system command is used to read a one-sector program from track 0, sector 1 of the specified disk drive. The load command accepts one optional parameter to specify from which physical disk to load. If this optional parameter is omitted, the load is from physical drive unit 0 (drive A). Floppy disk configurations have valid disk parameters of A through D. The usual load-from-disk-command for floppy drives will be L or LA, to load from drive A. Rigid disk configurations can have valid disk parameters of A through H.

> **Note**: The drive that is loaded from becomes logical drive A. Thus, when the operating system is loaded from drive E on the rigid (the LE command), physical disk E will be referenced as A and the physical floppy A is referenced as logical drive E.

The disk loader reads the first logical sector into memory at location 80h and starts execution at that address. Normally, the program will be a small loader that in turn reads in a larger program. This two-level bootstrap process makes the boot command application independent. The only requirements are that the first sector of the disk be reserved for a loader, the first byte of this loader not be an E5h, and the first 256 bytes of memory not be overwritten by the program being loaded.

11) R - READ DISK SECTOR COMMAND

This command allows one physical sector to be read from the specified disk drive to a designated address in memory (must be above 66H). The drive unit is a number between 0 and F hex, with 0 corresponding to physical drive A.

It should be noted that different disks may not have the same sector size. The read sector command will always read one physical sector, no matter what its length. Typically, single density disks will have 128-byte sectors, and double density disks will have 256-byte sectors. The sector size for rigid disk drives will always be 256 bytes or larger. Even though one physical sector is read, 256 bytes will be displayed after the read. Thus, when reading single density disks, only the first 128 bytes of the 256 bytes displayed on the screen are valid.

Physically, all floppy disks used with the 820-II and 16/8 begin with sector #1. However, when using the Monitor's R command, the first physical sector is accessed by specifying sector 0 in the command line.

12) W - WRITE DISK SECTOR COMMAND

A "**W**" and a **<return>** is required with the Write Disk Sector command before parameters will be accepted. A second "**W**" and parameters and a **<return>** is then required. The write disk sector command allows one physical sector to be written to the specified disk drive from the designated address in memory. The drive unit is a number between 0 and F hex, with 0 corresponding to physical drive A.

Note that different disks may not have the same physical sector size. The write sector command will always write one physical sector, no matter what its length. In general, single density disks will have 128-byte sectors, and double density disks will have 256-byte sectors. The sector size for the rigid disk drive will also be 256 bytes.

Physically, all floppy disks used with the 820-II and 16/8 begin with sector #1. However, when using the Monitor's W command, the first physical sector on double density and rigid disks is accessed by specifying sector 0 in the command line.

13) B - BAUD RATE COMMAND

This command sets the baud rate for the designated serial I/O channel, (printer or communications port). An optional parameter is required to change the baud rate from the default (1200-Printer, 300-Comm). If a second optional parameter is not specified, then the baud rate is set for the printer port (channel B). Specifying channel A will cause the designated or default baud rate to be set for the communications port. At power-on or reset, both serial ports are set for 7 data bits and even parity with one stop bit. See also the Host Terminal Mode section.

14) T - TYPEWRITER COMMAND

In Typewriter Mode, any information typed on the standard 96-character keyboard will be sent to the serial printer port.

To use Typewriter Mode, type the following parameters (in bold):

```
T   or  T#

        T = Typewriter Mode
        # = Baud Rate (1200)
```

```
For example:

    T     Typewriter Mode
    T5    Typewriter Mode 300 Baud (from Chart, Appendix K-1)
```

15) H - HOST TERMINAL

```
Host Terminal          H (channel) (baud rate) (data bits) (parity)
                       (stop bits)
```

The Xerox 820-II or 16/8 may be used as a terminal to a Host. By typing an **H** at power-on, the firmware initializes software in ROM that permits communication with a host computer.

Options (default settings are bolded):

Channel	**A** (communications port) B (printer port)
Baud rate	Channel A - **300** Channel B - **1200** (For other Baud Rate options, see Appendix K)
Data bits	**7** or 8
Parity	Odd, **Even**, or None
Stop bits	**1** or 2

For example, typing

```
H<space>B<space>8<space>None<space>2<return>
```

would put your system in Host Mode with 2 stop bits, no parity, 8 data bits, at 1200 baud on the Printer port.

Options may be altered using the Monitor Output command ***before*** typing an **H <return>** to load host terminal mode. It is important to note that these settings will remain in effect until the the system is turned off, the RESET button in the rear is pressed, or a disk is loaded that has had the CONFIGUR program run on it.

To change to odd parity:

O06 04	meaning	**O**utput to the SIO Channel A control port **(06)** selecting internal register 4 **(04)**
O06 45		**O**utput a **45**h to the SIO Channel A control port **(06)** which sets internal SIO register 4 to enable odd parity

To change to no parity:

O06 04	**O**utput to the SIO Channel A control port **(06)** selecting internal register 4 **(04)**
O06 44	**O**utput a **44**h to the SIO Channel A control port **(06)** which sets internal SIO register 4 to enable no parity

To change to 8 data bits (receiver and transmitter):

O06 03	**O**utput to the SIO Channel A control port **(06)** selecting internal register 3 **(03)**
O06 C1	**O**utput a **C1**h to the SIO Channel A control port **(06)** which sets internal SIO register 3 to 8 data bits for the receiver
O06 05	**O**utput to the SIO Channel A control port **(06)** selecting internal register 5 **(05)**
O06 EA	**O**utput an **EA**h to the SIO Channel A control port **(06)** which sets internal SIO register 5 to 8 data bits for the transmitter

For example, to change to no parity, 8 data bits, and set the baud rate to 1200, the following parameters should be entered at power-on:

O06<space>04<return>

O06<space>44<return> sets **no parity**

O06<space>03<return>

O06<space>C1<return> sets **8 data bits receiver**

O06<space>05<return>

O06<space>EA<return> sets **8 data bits transmitter**

B<space>07<space>A<return> sets **1200 baud, comm port**

H<return> loads Host Mode with the above parameters

Host Terminal Command Set

Host mode has a command set that can be used by pressing the <CTRL> key and one of the **Numeric Pad** keys. Note: Scroll up and Scroll down (↑ and ↓) do not require the <CTRL> key when using the 16/8.

<CTRL> +	**Meaning**
↑	**Scroll up.** Up-arrow scrolls up text on the screen with wrap around.
↓	**Scroll down.** Down-arrow scrolls down text on the screen with wrap around.
DEL	**Enable local echo.** Characters typed on the keyboard are displayed on the screen and transmitted through the serial port. Touching <CTRL> + DEL again disables local echo mode.
Line Feed	**Enable local auto line feed.** When <return> is touched, a line feed is sent to the local screen display but not transmitted through the serial port. Touching <CTRL> + LF again disables local line feed mode.
1	**Enable remote echo.** Characters received through the serial port are echoed back to the transmitting device. In this mode, the 820-II or 16/8 may act as a host to another terminal. Touching <CTRL> + 1 again disables remote echo mode.
2	**Enable remote auto line feed.** Carriage return codes received through the serial port are echoed to the remote device as carriage return/line feed codes. Touching <CTRL> + 2 again disables remote auto line feed.
(period)	**Transmit BREAK.** When <CTRL> and the period key on the numeric keypad are touched, a break condition is enabled on the serial port until: 1. <CTRL> + . is touched again 2. Any other character is typed.

<CTRL> +	Meaning
	"Toggling" the break function allows the length of the break condition to be determined by the user. Some host computers require a very short break condition, while some communications control devices require a long break condition.
ESC	**Exit Host Terminal Mode.**

Note: In Host Mode, the 820-II or 16/8 will respond to the special Display Control Codes listed in the CRT Control & Interface section.

16) P - PROTOCOL COMMAND

The protocol command alters the method used to control the transmission of characters to the printer (for different types of serial printers). Normally, XON/XOFF protocol is enabled to allow efficient communications with a Xerox 20 or 40 CPS printer. Since this is a "transparent" protocol, it will not interfere with printers that don't use XON/XOFF.

The protocol command requires at least one parameter to enable or disable the XON/XOFF protocol. P1 enables this protocol, while P0 disables it.

A second type of protocol is used for printers that control the transmission of characters by means of "reverse channel" or other hardware signals. Two signals may be used to control the transmission of characters to the printer:

CTS (Clear To Send)	Printer connector Pin 5
DTR (Data Terminal Ready)	Printer connector Pin 20

Two parameters are used to specify how these signals will be used for "hardware handshaking"; the first designates which signals are to be checked, and the second indicates which logical state will be used to enable the transmission of data.

The most commonly-used modes are shown below. The voltage level is the EIA RS- 232 level measured at the printer connector:

P1<space>28<space>28<return>

Check CTS and DTR, pins 5 and 20. If either changes to false (-12), stop transmission.

P1<space>8<space>8<return>

Check DTR, pin 20. If false (-12), stop transmission.

P1<space>20<space>20<return>

Check CTS, pin 5. If false (-12), stop transmission.

The following examples show the values for some less-common printers that require transmission be stopped with signals of the opposite sense. Notice these examples also enable the XON/XOFF protocol by specifying a **1** as the first parameter.

P1<space>28<space>0<return>

Check CTS and DTR, pins 5 and 20. If either changes to true (+ 12), stop transmission.

P1<space>8<space>0<return>

Check DTR, pin 20. If true (+ 12), stop transmission.

P1<space>20<space>0<return>

Check CTS, pin 5. If true (+ 12), stop transmission.

Notes

Operating System Interface

The preferred method of accessing the resources of the 820-II and 16/8 is through one of the operating systems (CP/M-80, CP/M-86, or MS-DOS). The operating system functions available are documented in the manuals listed below.

CP/M-80	Interface Guide section of Digital Research's CP/M 2.2 Operating System Reference Manual.
CP/M-86	Digital Research's CP/M-86 User's Guide, System Guide, and Programmer's Guide.
MS-DOS	MS-DOS Programmers Guide.

Accessing CP/M-80 and CP/M-86 BIOS
CP/M-80/CP/M-86 also provide a BIOS (Basic Input/Output System) interface that is available to the programmer. The BIOS interface is described in the following manuals:

CP/M-80	Alteration Guide section of Digital Research's CP/M 2.2 Operating System Reference Manual.
CP/M-86	Digital Research's CP/M-86 User's Guide, System Guide, and Programmer's Guide.

CP/M-86 has an operating system function (#50) that provides access to the CP/M-86 BIOS.

The BIOS interface for CP/M-80 version 2.2 is not supported as an operating system function. An application program may call 16 of the 17 BIOS vectors; the first vector Cold boot may not be called. Because the BIOS jump table is not anchored to any fixed memory locations, application programs must not directly call any of the jump vectors without first calculating the address of the desired vector. At address 0000H is a jump instruction to the **second** BIOS vector -**wboot**. The application program should read the address stored at address 0001H and 0002H, then add the offset of the desired BIOS jump vector and call this "calculated" address.

For example, suppose an application program needed to determine whether or not the list device is busy. This is not supported with an operating system function call under CP/M-80 2.2.

```
        ;Users program

        call    biolsts         ;get status of list device
        or      a               ;result is returned in a
                                ;00    = not ready
                                ;else  = ready

        ;Continue

Biolsts:

        ld      hl,(0001H)      ;Get address of wboot
        ld      l,15*3          ;15th vector 3 bytes per vector
        jp      (hl)
```

The reason that the biolsts label was "called" from the main program is to put a return address on the stack. Remember, all BIOS routines end with a return instruction.

Additional BIOS Information

The following describes parameters for some of the BIOS functions that are not described in the Alteration Guide.

Sectran - The sector translate vector is documented to receive a logical sector number in the **BC** register pair, and the address of a logical-to-physical translate table in the **DE** register pair, returning the physical sector number from the table in the **HL** register. In the 820-II and 16/8, when a double density disk or a rigid disk is being accessed, the **DE** register pair contains a 0000H indicating no logical-to-physical skew table. When this occurs, the logical sector number is returned in the **HL** register.

Seldsk - If bit 0 of the **E** register is 0, the BIOS recognizes this as a first-time select of the disk and will request the physical disk driver to determine the type of media currently in the drive.

Write - The **C** register contains the write type.

0	=	Write to an allocated data block
1	=	Write to directory
2	=	Write to an unallocated data block

CP/M Logical - 820-II Physical Device Mapping

The IOBYTE has been partially implemented in the 820-II to enable optional re-assignment of CP/M character devices (console and list) to different physical devices on the 820-II (CRT/keyboard, serial modem port, serial printer port, and parallel printer port). This logical-to-physical device mapping can be changed either under program control or with CP/M's transient command, STAT.

CP/M Logical device names	Physical device names			
CON:	TTY:	CRT:	BAT:	UC1:
RDR:	TTY:	PTR:	UR1:	UR2:
PUN:	TTY:	PTP:	UP1:	UP2:
LST:	TTY:	CRT:	LPT:	UL1:

The chart above lists the CP/M logical device names in the left column and the valid physical devices for each logical device is listed to the right of the logical device name. For example the logical console device can be mapped to the physical TTY:, CRT:, BAT:, or UC1:, but not PTR:.

The chart below shows the mapping of physical device names to physical devices on the 820-II.

CP/M physical device names	820-II Physical devices
BAT:	Serial printer port
CRT:	820-II CRT and keyboard
LPT:	Serial printer port
PTP:	Serial modem port
PTR:	Serial modem port
TTY:	Serial modem port
UC1:	Serial printer port
UL1:	Parallel printer port
UP1:	Serial modem port
UP2:	Serial modem port
UR2:	Serial modem port
UR1:	Serial modem port

CP/M Logical - 16/8 Physical Device Mapping

The IOBYTE has been fully implemented on the 16/8. The tables below describe the logical to physical device mapping for CP/M-80 and CP/M-86.

CP/M-80 Logical device names	**Physical device names**			
CON:	TTY:	CRT:	BAT:	UC1:
RDR:	TTY:	PTR:	UR1:	UR2:
PUN:	TTY:	PTP:	UP1:	UP2:
LST:	TTY:	CRT:	LPT:	UL1:

CP/M-86 Logical device name	**Physical device names**			
CON:	TTY:	CRT:	BAT:	UC1:
AXI:	TTY:	PTR:	UR1:	UR2:
AXO:	TTY:	PTP:	UP1:	UP2:
LST:	TTY:	CRT:	LPT:	UL1:

CP/M physical device names	**16/8 Physical devices**
BAT:	Serial printer port
CRT:	16/8 CRT and keyboard
LPT:	Serial printer port
PTP:	16/8 CRT and keyboard
PTR:	16/8 CRT and keyboard
TTY:	Serial modem port
UC1:	Inter-processor communication channel
UL1:	Parallel printer port
UP1:	Serial printer port
UP2:	Inter-processor communication channel
UR2:	Inter-processor communication channel
UR1:	Serial printer port

In the 16/8 configuration, an application program running on the 8086 can communicate with an application program running on the Z80-A through the inter-processor communication channel by changing the I/OBYTE value and using console input and console output functions.

INPUT		
A0	=	Stop 8086
A1	=	Start 8086
OUTPUT		
A0, D7	=	1 : Lock 8086

I/O PORT ASSIGNMENTS

Note: These input/output ports are accessible by the Z80-A only. The 8086 microprocessor on the 16/8 cannot access these I/O ports.

Port # (hex)	**Assignment**
00	Channel A Baud Rate (Modem) (write only)
01	Channel A Baud Rate (Modem) (write only)
02	Channel A Baud Rate (Modem) (write only)
03	Channel A Baud Rate (Modem) (write only)
04	SIO Channel A (Modem) Data
05	SIO Channel B (Printer) Data
06	SIO Channel A (Modem) Control
07	SIO Channel B (Printer) Control
08	GP-PIO Channel A Data
09	GP-PIO Channel A Control
0A	GP-PIO Channel B Data
0B	GP-PIO Channel B Control
0C	Channel B Baud Rate (Printer) (write only)
0D	Channel B Baud Rate (Printer) (write only)
0E	Channel B Baud Rate (Printer) (write only)
0F	Channel B Baud Rate (Printer) (write only)
10	Floppy Disk Controller Status/Command Register Fixed Disk PIO Channel A Data
11	Floppy Disk Controller Track Register Fixed Disk PIO Channel A Control
12	Fixed Disk PIO Channel B Data Floppy Disk Sector Register
13	Fixed Disk PIO Channel B Control Floppy Disk Data Register
14	CRT Scroll Register (write only)
15	CRT Scroll Register (write only)
16	CRT Scroll Register (write only)
17	CRT Scroll Register (write only)
18	CTC Channel 0
19	CTC Channel 1
1A	CTC Channel 2

I/O PORT ASSIGNMENTS continued

Port # (hex)	Assignment
1B	CTC Channel 3
1C	System PIO Channel A Data
1D	System PIO Channel A Control
1E	System PIO Channel B Data (keyboard)
1F	System Pio Channel B Control (keyboard)
(20-27 not used and not available)	
28	Speaker cone push (write only)
29	Speaker cone pull (write only)
(2A-2F not used and not available)	
30	Select Single Density
31	Select Double Density
(32-33 not used and not available)	
34	Reset CRT Font Generator to ROM #1 (write only)
35	Reset CRT Font Generator to ROM #2 (write only)
36	Set Low-Light Video Mode (write only)
(37-67 not used and not available)	
68	Asynchronous Communications (write only)
69	Synchronous Communications (write only)
80 - 9F	Reserved
FE - FF	Reserved
A0 - A3	16/8 CPU Board
A4 - AF	Reserved
B0 - BF	Reserved

ROM Operating System Interface

The 820-II and 16/8 also provide a series of ROM operating system jump vectors that can be accessed by a program executing on the Z80-A for other functions available on the 820-II and 16/8. It is important to note that these should be used only when the necessary service is not provided by the operating system. Use of these ROM services makes the program un-transportable to most other computers. Also, use of the ROM operating system I/O services may make the program inoperable under the dual CP/M-80/86 system.

CRT Overview

The CRT functions involve the moving of characters to the CRT RAM and character display. (The entry points described in this section are in the Monitor; see IOBYTE, starting on page 167 for BIOS display-related calls). CRTOUT simply displays a character at the cursor position and increments the cursor. FASTCRT also displays a character at the cursor position, but keeps track of and returns information about characters lost at the end of a line, deleted characters, etc. SETCUR stores a CRT RAM address; OUTCUR then stores a character at this address. CRTLDIR will move a block of memory (or group of characters to or from CRT RAM).

CRT Output

Entry Point:	F00CH
Function(s)	At the current cursor position, display the character in register A or perform the special function defined by the character sequence supplied in consecutive calls to CRTOUT.
Arguments:	(A) = the character to display
Value(s) Returned:	None
Registers Saved:	All
Errors Returned:	None

Fast CRT Output

Entry Point:	F00FH
Function(s):	At the current cursor position, display the character in register C or perform the special function defined by the character sequence supplied in consecutive calls to FASTCRT.
Arguments:	(C) = the character to display
Value(s) Returned:	
Normal display character:	(A) = character under the cursor (HL) = CRT RAM address of the cursor

The special functions will return the following:

Character Insert 1B 51h	(A) = character that was lost off the end of the line
Character Delete 1B 57h	(A) = character that was deleted
Line Insert 1B 45h	The line that was lost off of the bottom of the screen is moved to the Command Processor's line buffer. This buffer is located immediately after the Time-of-Day clock variables, whose address is obtained by calling F039.
Line Delete 1B 52h	The line that was deleted is moved to the line buffer as in Line Insert.
Line Feed 0Ah	The A register returns a flag indicating whether or not the line feed caused the top line to be lost (scrolled) (A = 0 - scrolled, A ≠ 0 - not scrolled). If so, the line may be found in the line buffer as in Line Delete.
Registers Saved:	None
Errors Returned:	None

Set Direct CRT Cursor

Entry Point:	F02DH
Function(s):	Store the address passed in registers HL for use in successive calls to Direct CRT Display.
Arguments:	(HL) = CRT RAM address
Value(s) Returned:	None
Registers Saved:	None
Errors Returned:	None

Direct CRT Display

Entry Point:	F030H
Function(s):	Store the character in C in the CRT RAM at the Direct Cursor location. The normal cursor is unaffected. The direct cursor address is incremented, however line/screen overflow is not processed.
Arguments:	(C) = character to display (HL) = CRT RAM address
Value(s) Returned:	None
Registers Saved:	None
Errors Returned:	None

CRT Memory Block Move

Entry Point:	F033H
Function(s):	This entry point moves a memory block to/from or within the alternate memory bank. It functions like the Z80-A LDIR instruction except that it also takes care of switching memory banks. When data is transferred between Bank 0 and Bank 1, source data is first moved to the internal line buffer, then the memory bank is switched and the data saved in the internal buffer is transferred to its destination. This sequence of operations is repeated until all source data has been transferred. When data is transferred within the CRT RAM, there is no internal buffering performed.
Arguments:	(HL) = source address (DE) = destination address (BC) = number of bytes to move (A) = type of move = 0 if move CRT RAM to CRT RAM < 0 if move system RAM to CRT RAM (i.e., A = FFH) > 0 if move CRT RAM to System RAM (i.e., A = 1)
Value(s) Returned:	HL, DE, BC updated as in LDIR instruction
Registers Saved:	None
Errors Returned:	None

Execute Physical Driver

This entry point is the heart of the disk system. Upon entry, register HL must point to a nine-byte block of memory called the Physical Driver Request Block (PDRB) which must be formatted as shown below:

```
00:   db   command    ;FF =  Select
                      ;00 =  Write
                      ;01 =  Read
01:   ds   1          ;for system use
02:   db   Ldrive     ;Logical Drive for request (00 - 0F)
03:   dw   Track      ;Track number for request
05:   dw   Sector     ;Sector number for request
07:   dw   Address    ;Address of sector buffer for request
```

The byte holding the Logical Drive (HL + 02) is used to select the appropriate physical disk driver by indexing into the Select Table to obtain the driver unit as well as the driver entry point address. Byte (HL + 01) is filled with the physical unit number for this physical driver, then control is passed directly to the physical disk driver.

User-written disk drivers may be linked into the Select Table if these drivers conform to the virtual interface described. The following command values (HL + 00) must be supported by any user generated physical driver.

Entry Point:	F02AH
Function(s):	**FF - Select Media Format** - This command causes the disk driver to identify the media in the physical unit. Registers HL return pointing to a CP/M-compatible Disk Parameter Header if the media was successfully identified. Otherwise HL contains zero. This command may cause several disk accesses because it must determine the disk's density and the number of sides. Therefore, it should not be issued repeatedly, or system performance may be affected. Xerox's CP/M

issues this command whenever a disk drive is 'logged in'.

00 -- Write Sector - This command causes the physical sector identified by bytes 03 through 06 of the PDRB to be written from the buffer addressed by bytes 07 and 08. The acceptable values for Track and Sector vary with different physical disk drivers.

01 -- Read Sector - This command causes the physical sector identified by bytes 03 through 06 of the PDRB to be read into the buffer addressed by bytes 07 and 08. The acceptable values for Track and Sector vary with different physical disk drivers.

Note: On read/write sector requests, the first physical sector number on double density floppies is actually 1; however, the PDRB must request sector 0 for the first physical sector. The second physical sector can be accessed with a request for sector 1, etc. Single density floppies access physical sector 1 by requesting sector 1.

Arguments:	HL	=	address of Physical Driver Request Block (PDRB)
Value(s) Returned:			
If SELECT command:	HL	=	address of a CP/M-compatible Disk Parameter Header if the media was successfully identified
	HL	=	0 otherwise
If READ or WRITE command:	(A)	=	00 if no error
	(A)	=	FF if error
Registers Saved:	None		

Printer Overview

The Monitor printer entries SIOST, SIOIN, SIOOUT, and SIORDY are those functions which check status and provide for input and output for SIO Channel B, a serial printer. Printer protocols are processed only by the SIO Channel B entries.

SIO-B Input Ready Status

Entry Point:	F012H
Function(s):	Get SIO Channel B input ready status.
Arguments:	None
Value(s) Returned:	(A) = 00 if no data available (A) = FF if data available
Registers Saved:	All except AF
Errors Returned:	None

SIO-B Input Data

Entry Point:	F015H
Function(s):	Get SIO Channel B input character. If an input character is not ready, IDLE is called repeatedly until one is ready.
Arguments:	None
Value(s) Returned:	(A) = character
Registers Saved:	All except AF
Errors Returned:	None

SIO-B Output Data

Entry Point:	F018H
Function(s):	Wait until the SIO Channel B transmitter is ready (by calling SIORDY), then transmit the character in (A). IDLE is called while the transmitter is not ready.
Arguments:	(A) = character to transmit
Value(s) Returned:	None
Registers Saved:	None
Errors Returned:	None

SIO-B Output Ready Status

Entry Point:	F03FH
Function(s):	Determine if the device connected to SIO Channel B is ready to receive data. SIORDY supports the configured printer protocol and the DC1/DC3 (XON/XOFF) sequence.
Arguments:	None
Value(s) Returned:	(A) = 00 if not ready (A) = FF if ready
Registers Saved:	All except AF
Errors Returned:	None

Communications Overview
Monitor entries for status and input/output are also provided for SIO Channel A, which is generally used as a communications port. These entries are COMINS, COMINP, COMOUT, AND COMOTS.

Communications Input Ready Status

Entry Point:	F05AH
Function(s):	Get the SIO Channel A input ready status.
Arguments:	None
Value(s) Returned:	(A) = 00 if not ready
	(A) = FF if ready
Registers Saved:	All except AF
Errors Returned:	None

Communications Input Data

Entry Point:	F05DH
Function(s):	Input character from SIO Channel A.
Arguments:	None
Value(s) Returned:	(A) = character
Registers Saved:	None
Errors Returned:	None

Communications Output Status

Entry Point:	F060H
Function(s):	Determine if the SIO Channel A transmitter is ready to accept data.
Arguments:	None
Value(s) Returned:	(A) = 00 if ready
Registers Saved:	All except AF
Errors Returned:	None

Note: IDLE is not called by the Channel A drivers. Therefore, these entries may be called by a user-written IDLE procedure. In this manner, you may drive Channel A while other I/O (disk, printer, etc.) is pending.

Keyboard Overview

The Monitor keyboard entries, KBDST and KBDIN, provide for keyboard status and input. A 16 (decimal) key type-ahead FIFO is maintained for the keyboard on an Etch 2 CPU.

Keyboard Status

Entry Point:	F006H
Function(s):	Determine if a keystroke is available.
Arguments:	None
Value(s) Returned:	(A) = 00 if no character available
	(A) = FF if character available
Registers Saved:	All except AF
Errors Returned:	None

Keyboard Input

Entry Point:	F009H
Function(s):	Wait for keyboard input data. IDLE is called while input is not available.
Arguments:	None
Value(s) Returned:	(A) = character
Registers Saved:	All except AF
Errors Returned:	None

IOBYTE Directed I/O

The IOBYTE function allows for physical-to-logical device mapping. This mapping capability provides flexibility and device isolation for the user. If IOBYTE-directed I/O is used, a program does not have to know which devices are currently active; the Operating System will perform the logical-to-physical I/O mapping.

The mapping is based on the contents of IOBYTE, location 0003, which defines the assignment of devices to the CONSOLE, READER, PUNCH, and LIST devices. Monitor entries IOCONS, IOCONI, and IOCONO provide for status of and input/output to the CONSOLE device. IOLIST and IOLSTS provide for status of and output to the LIST device. In addition, the BIOS has entries PUNCH and READER to access the Communications Channel.

Console Status through IOBYTE

Entry Point:	F04BH
Function(s):	Get status of the assigned CONSOLE device by dispatching request based upon the current (bit) values of IOBYTE as follows: 00 - Comins 01 - Kbdst 10 - Siost 11 - Siost
Arguments:	None
Value(s) Returned:	(A) = 00 if not ready (A) = FF if ready
Registers Saved:	None
Errors Returned:	None

Console Input through IOBYTE

Entry Point:	F04EH
Function(s):	Get input from the assigned CONSOLE device by dispatching request based upon the current (bit) values of IOBYTE as follows: 00 - Cominp 01 - Kbdin 10 - Sioin 11 - Sioin
Arguments:	None

Value(s) Returned: (A) = character
Registers Saved: None
Errors Returned: None

Console Output through IOBYTE
Entry Point: F051H
Function(s): Send output to the assigned CONSOLE device by dispatching request based upon the current (bit) values of IOBYTE as follows:
- 00 - Comout
- 01 - Fastcrt
- 10 - Sioout
- 11 - Comout

Arguments: (C) = character to transmit
Value(s) Returned: None
Registers Saved: None
Errors Returned: None

Printer Output through IOBYTE
Entry Point: F054H
Function(s): Send output to the assigned LIST device by dispatching request based upon the current (bit) values of IOBYTE as follows:
- 00 - Comout
- 01 - Fastcrt
- 10 - Sioout
- 11 - Pioout (parallel printer)

Arguments: (C) = character to transmit
Value(s) Returned: None
Registers Saved: None
Errors Returned: None

Printer Status through IOBYTE

Entry Point:	F057H
Function(s):	Get status of the assigned LIST device by dispatching request based upon the current (bit) values of IOBYTE as follows: 00 - Comots 01 - Return ready 10 - Siordy 11 - Piosto (parallel printer)
Arguments:	None
Value(s) Returned:	(A) = 00 if not ready (A) = FF if ready
Registers Saved:	All except AF
Errors Returned:	None

Programmable Functions
The following are system exit points. They are provided to allow application-specific activities on a 1-second interrupt, when an I/O request is pending or when a soft disk error occurs.

In order for the ROSR Monitor to call these functions, the address of your routine must be patched into the appropriate Monitor vector table entry. A vector table entry is in the following format (LSB/MSB means Least/Most Significant Byte of address):

Byte 1	Byte 2	Byte 3
Jump Instruction	LSB	MSB

For example, a jump to location F048H would look like:

Byte 1	Byte 2	Byte 3
C3	48	F0

The application must first retrieve and save the address portion of the appropriate vector table entry. (The saved address must be restored when the application terminates.) After the vector table contents are saved, the application must store the address of its programmed function into the vector table entry (overwriting the previous contents). You must be careful to replace only the address portion of the jump instruction and to put the address bytes in the proper order.

A sample code sequence for patching (using Z80-A assembly language):

Example

```
        .z80
Secvec  equ     0f048h          ;accessible 1-second interrupt vector
Commrm  equ     0c000h          ;start of common memory
Syspio  equ     1ch             ;system pio data
Banksw  equ     7               ;bank switch bit

        aseg
        org     100h

;--     Initialization
;
Start:  ld      hl,(6)          ;get highest available address under bdos
        ld      de,usrrou+usrsiz
        or      a
        sbc     hl,de           ;if end of driver > highest available address
        jp      c,erext         ;then exit with error message
        ld      hl,usrst        ;if there is enough space, move user's routine
        ld      de,commrm       ;to common memory.
        ld      bc,usrsiz
        ldir
        call    swap            ;swap clock vector with value at swpvec

;--     Now the accessible 1-second interrupt has been changed
;       to jump to the routine contained in the application program.
;       The remainder of the application program would be here branching
;       to the exit routine when it is time to go back to the operating
;       system.
;

                ;********************************
                ;               *
                ;*   Your Program goes here   *
                ;               *
                ;********************************

;--     On exit, swap clock vector again to restore original value.
;
Exit:   call    swap            ;swap vectors back
        jp      0               ;return to operating system
```

```
Erext:    ld      de,ermsg        ;give error message
          ld      c,9             ;print string function
          call    5
          jp      0

Usrst     equ     $

          .phase  commrm
;--        This routine is executed as part of the 1-second interrupt
;          service routine. It is possible for the CRT bank to be
;          enabled; this routine forces the RAM bank to be selected.
;          Also, because this is executed as part of an interrupt service
;          routine, the interrupts do not need to be disabled during bank
;          switching.
;
Usrrou:   in      a,(syspio)      ;read current value of system port
          push    af              ;& save current value away
          res     banksw,a        ;force bank switch to ram bank
          out     (syspio),a
          ld      hl,(clkval)     ;get current value of clkval
          inc     hl              ;increment & save new value
          ld      (clkval),hl
          ld      hl,(swpvec)     ;hl = previous address of clock vector
          pop     af              ;get previous value of system port
          out     (syspio),a      ;& restore it
          jp      (hl)            ;instead of returning so clock routines can
                                  ;be chained

          .dephase

Usrsiz    equ     $-usrst

;--        Swap address at location Swpvec with address in jump vector
;
Swap:     ld      hl,secvec+1     ;point hl to one second vector
          ld      bc,(swpvec)     ;bc = address new value for clock vector
          di                      ;disable interrupts while changing vector
          ld      e,(hl)
          ld      (hl),c
          inc     hl
          ld      d,(hl)
```

```
        ld      (hl),b              ;Clock vector = users Clock routine
        ld      (swpvec),de         ;Save original vector for exit
        ei                          ;interrupts ok now
        ret

Clkval: dw      0                   ;Flag to be checked by program
Swpvec: defw    usrrou              ;Save system's 1-second vector here for exit

Ermsg:  db      'Not enough Space in common memory for program$'

        end
```

Accessible 1-second interrupt

Entry Point:	F048H
Function(s):	This exit point is called by the real time clock interrupt service routine once each second. This user-programmed function must follow the rules of interrupt service processing. Only registers HL and AF may be used. Any other registers must be saved/restored on the five level stack provided. The address of your 1-second interrupt processing routine must be patched at F049-F04A. Your routine should terminate with a jump to the address that was in this vector table entry (F049-F04A) prior to patching. Currently, this function simply returns.
Arguments:	HL = 16-bit seconds counter
Value(s) Returned:	N/A
Registers Saved:	The service routine must preserve all registers
Errors Returned:	N/A

Processing While I/O Pending

Entry Point:	F066H
Function(s):	This exit point is called by SIOIN, SIOOUT, KBDIN, and by the WD1797 and SA1403 disk drivers when an I/O request cannot be satisfied because the device is busy or not ready. It provides the capability of performing other activities while waiting for I/O. This entry must function as an interrupt service routine. That is, it must: switch to a local stack (no system stack space may be used), save ALL modified registers (including flag register), perform its function, restore all saved registers, and enable interrupts. The address of the idle processing routine must be patched at F067- F068. (When complete, the idle processing routine should jump to the contents of the vector table entry, F067-F068, prior to patching. This has effect of chaining all idle processors together and ensures that

each one has an opportunity to execute.) The original contents of this vector should be restored when the application completes. Currently this function simply returns.

Arguments:	N/A
Value(s) Returned:	N/A
Registers Saved:	You must save all registers
Errors Returned:	N/A

Soft Error Recording

Entry Point:	F069H
Function(s):	This exit point is called by the WD1797 and SA1403 disk drivers when a soft error occurs. It provides you with the opportunity to record and/or process occurrences of soft errors. The address of your soft error processing routine must be patched at F06A-F06B. The address of the idle processing routine must be patched at F067- F068. (When complete, the idle processing routine should jump to the contents of the vector table entry, F067-F068, prior to patching. This has the effect of chaining all idle processors together and ensures that each one has an opportunity to execute.) The original contents of this vector must be restored when the application completes. Currently this entry returns a non-zero condition.
Arguments:	**For WD1797:** HL = address of Physical Driver Request Block (for Xqdvr) (B) = current retry down counter **For SA1403:** HL = address of Physical Driver Request Block (for Xqdvr) command block + 1 (B) = current retry down counter
Value(s) to be Returned by You:	(B) = number of retries desired If (B) = 1 (no more retries) (A) = 00 and Z(ero) Flag Set no error returned to CP/M = FF and Z(ero) Flag Reset error returned to CP/M Currently, (B) is unchanged (A) = FF and Z(ero) Flag is Reset
Registers Saved:	All except AF
Errors Returned:	None

Miscellaneous Functions

Other Monitor entries are COLD which provides a software reset and WARM which is an exit point from the system. GETSEL will return the address of the logical-to-physical disk mapping table. DAYTIM returns the address of the timer variables. CONFIG returns the system configuration. SSP will initiate a screen print.

Cold Start

Entry Point:	F000H
Function(s):	This entry point may be called at any time to cause a Software Reset. The system is reloaded from ROM and all I/O devices are re-initialized.
Arguments:	None
Value(s) Returned:	None
Registers Saved:	SP, HL, (SP) are saved at FFE0 in RAM
Errors Returned:	No Return

Warm Start

Entry Point:	F003H
Function(s):	This system exit point is called by the keyboard interrupt service routine when <CTRL> + <ESC> is touched. When executing the power-on commands, (Typewriter, Host Terminal, etc.), this exit point is set to the address of the Command Processor Line Scanner. Thus, <CTRL> + <ESC> is used to exit the various command processors. When the L(oad) command enters the boot loader, it directs the exit point to the Cold Start entry point. This causes <CTRL> + <ESC> to act similar to pressing the RESET button. You may load the address of your own software-abort routine into locations F004-F005. This routine <u>must</u> be located in the upper 16k of RAM (above C000). Only the HL, BC, and AF registers are available for use if the routine RETurns to the keyboard interrupt driver. Any other registers used must be saved. Only five levels of stack space are available. All

rules of interrupt service processing must be followed. For example, no calls may be made to the system I/O drivers. Typically, a routine will set an abort flag that is monitored by the application and then exit with a return instruction. When the application sees the flag set, it should proceed with its own abort sequence.

To disable this function, simply patch a return instruction at location F003 (overwriting the jump operation code). The original contents of this vector should be restored when your application completes.

Arguments: N/A
Value(s) Returned: N/A
Registers Saved: N/A
Errors Returned: N/A

Get Disk Map Table Address

Entry Point: F036H

Function(s): The address of the Logical to Physical disk mapping table (see Appendix C) is returned in registers HL. If register H is non-zero on entry, the table address is stored in the two-byte variable pointed to by HL. This allows easy access by high level programming languages. The table consists of two sections. The first section contains sixteen two byte entries - one for each logical CP/M drive. The first byte of each pair indicates which physical disk driver to activate for an I/O request; the second byte specifies which physical unit within that physical driver to access. These byte pairs may be carefully rearranged with other byte pairs in the table. They may even be removed or overwritten, but they must not be duplicated elsewhere in the table. The second part of the table holds the addresses of eight physical disk driver entry points. By convention, driver number 0 always returns an error. It is used to force Select Errors on undefined logical drives.

Driver number 1 controls all of the standard disk systems. Additional virtual disk drivers linked into this table, with appropriate values in the first section, may be accessed through the normal CP/M disk I/O facilities.

Arguments:	H	=	0
	- OR -		
	HL	=	address of integer variable where the disk map table address will be stored
Value(s) Returned:	HL	=	address of the disk map table If the H register was non-zero on entry, the address of the disk map table will also have been stored in the address contained in HL on entry to GETSEL.
Registers Saved:	None		
Errors Returned:	None		

Get Address of Time-of-Day Variables

Entry Point: F039H

Function(s): This entry is used to gain access to the timer variables maintained by the system. As in GETSEL, if register H is non-zero on entry, it is used as the address of an integer variable in which to store the result. In any case, HL holds the timer address on exit. The return address points into the following structure (numbers are decimal):

```
        Milsec:  ds  2        ;Location incremented by CTC1
                              ;      interrupt (if enabled)
                 ds  2        ;(unused)
        Ticker:  ds  2        ;Increments once per second
        Steprt:  ds  1        ;WD1797 step rate
        Motor:   ds  1        ;Disk Motor/Select timeout (1Hz)
    HL→ Day:     ds  1        ;01-31
        Month:   ds  1        ;01-12
        Year:    ds  1        ;80-99
        Hour:    ds  1        ;00-23
        Minute:  ds  1        ;00-59
        Second:  ds  1        ;00-59
        Linbuf:  ds  80       ;Line buffer referred to in
                              ;      FASTCRT and CRTLDIR
```

Arguments: H = 0

- OR -

HL = address of integer variable where the address of the time of day variables will be stored

Value(s) Returned: HL = address of the time of day variables If the H register was non-zero on entry, the address of the time of day variables will also have been stored in the addressed contained in HL on entry to DAYTIM.

Registers Saved: None

Errors Returned: None

Get Configuration Status

Entry Point:	F03CH		
Function(s):	This entry point returns current configuration. This function should be used to get the system Revision level and status information such as: what kind of disk system is present, the current keyboard mask state, or other variable information concerning the 820-II or 16/8. Only four status bits are currently defined (all zero bits are reserved for system use), but more may be added in later releases.		
Arguments:	H	=	0
	-OR-		
	HL	=	address of integer variable where the address of the configuration status will be stored
Value(s) Returned:	H	=	dvvvvvvv (Revision level)*100 - 400
	d	=	1 means CP/M-86 is loaded
		=	0 means CP/M-86 is not loaded
For example:	vvvvvvv is ROM Version # -400; i.e., 4.03 ROM returns the value 3 in bits 0-6		
Monitor level 4.03	(H)	=	(4.03)*100 - 400 = 3
	L	=	kddfL000 where:
	k	=	0 7-bit keyboard data
		=	1 8-bit keyboard data
	dd	=	00 Rigid disk not present
		=	10 8" 8-megabyte rigid disk present
	f	=	0 8" floppies present
		=	1 $5\frac{1}{4}$" floppies present
	L	=	1 Low Profile keyboard present
		=	0 ASCII keyboard present
	H	=	(revision level)*100 - 400
	L	=	configuration status If the H register was non-zero on entry, the configuration status will also have been stored in the address contained in HL on entry to CONFIG.
Registers Saved:	None		
Errors Returned:	None		

Start Screen Print

Entry Point:	F045H
Function(s):	This entry point initiates background screen print. Don't change the screen during printing or the printout won't be what you expect.
Arguments:	None
Value(s) Returned:	None
Registers Saved:	None
Errors Returned:	None

Documented System Storage and Structures

The documented system variables and structures include keyboard FIFO, available memory pointers, disk mapping and driver selection tables, disk command block and timer and clock variables. A listing of each of these variables and structures is provided in Appendix C.

SYSTEM DISPLAY

Modes of Operation
The display has two modes of operation, Display Character Mode and Graphics Mode.

The CPU Board is equipped with a CRT display controller for use with a video monitor as the system console output device. The refresh memory for the CRT is bank-switchable from the system's 64k byte memory space and includes a hardware address translation circuit for high speed scrolling. The Character Mode contains an output driver routine for the CRT that emulates the characteristics of a typical stand-alone video terminal. All character codes between 00 and 7F hex are directly displayable on the screen. Each character is formed in a 5x8 dot matrix. When the most significant bit of the character is set to "1", the attribute function is turned ON. One of three attributes may be chosen: Blink, Low Intensity, or Inverse Video. Only one attribute can be displayed at a time, and only those characters with the most significant bit set will show the selected attribute.

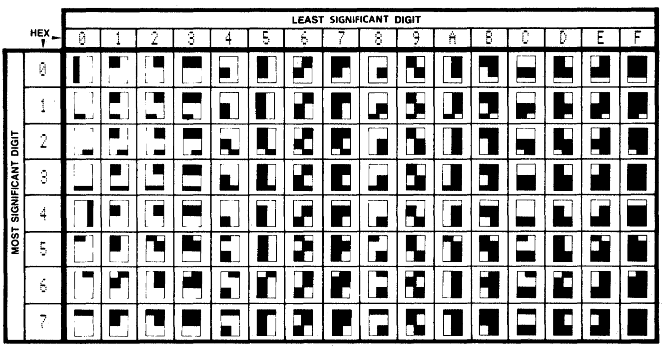
To display an up arrow (09h) with an attribute, output a 09h character using the CP/M function "Direct I/O" #6. A low-intensity up arrow (89h) is displayed like all the other codes described above.

For more information on the CRT, see pages 15 to 17.

Text Character set

HEX	LEAST SIGNIFICANT DIGIT															
MOST SIGNIFICANT DIGIT	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	□	↕	█	♦	§	½	¼	±	⇷	↑	↓	→	←	↕	Ø	⇸
1	°	²	º	▪	↓	¶	↑	µ	●	‖	↲	↲	0	⇥	⇤	0
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	[illegible]

Graphics Character Set



Programming Considerations

Display Character Mode

New characters are stored on the screen at the locations occupied by the cursor. The cursor is then moved one space to the right. If the cursor is positioned at a screen location occupied by a non-blinking character, the presence of the cursor will be indicated by making the overlaid character blink. If a line feed (0Ah = LF) is output when the cursor is on the bottom line of the screen, the entire display is scrolled up one line and a new blank line is created on the bottom. If the displayed character is output when the cursor is in the right-most column of the screen, an automatic carriage return and line feed are generated.

All characters codes between 20h and 7Fh are directly displayable on the screen. All character codes between 00h and 1Fh are interpreted as control characters. The video display may be controlled by these control codes and escape sequences to perform screen manipulations.

Display Manipulation through CTRL codes

CONTROL SEQUENCES

Code (hex)	Function
05	Set cursor character as next character
06	Restore previous attribute mode
07	Bell
08	Backspace or cursor left
09	Horizontal tab
0A	Line feed or cursor down
0B	Cursor up
0C	Cursor right
0D	Carriage return
11	Clear to end of screen
18	Clear to end of line
1A	Clear screen and home cursor
1B	Escape
1E	Home
1F	Display next character direct

DISPLAY CODE DESCRIPTION

The display control codes of the 820-II and 16/8 PCs are downwardly compatible with the original 820 with several advanced editing features added. The following summarizes the effect of each of the display codes.

CONTROL CODES

05h	**Set cursor character.** After receiving this code, the next character is interpreted as the code to be used as the cursor. Only codes between 0 and 20 hex will be accepted. The normal cursor code is 02h. The "space" character (20h) is a special case used to eliminate the display of a cursor. This is useful for displaying a screen without a large, visibly-moving cursor for special effects.
06h	**Restore previous attribute mode.** Whenever the attribute mode is changed, the previous mode is remembered. In this way, a program can set its own attributes for unique display requirements, and then restore the mode that was in effect before the program was run. Since the user may set a default attribute mode with CP/M's CONFIGUR program, it is desirable to restore the default mode after if has been temporarily changed.
07h	**Bell.** This code will sound a short tone to alert the operator.
08h	**Backspace or cursor left.** Moves the cursor one column position to the left without altering the character under the cursor.
09h	**Horizontal tab.** Moves the cursor to the next tab stop. Tabs are pre-set for every eighth column.
0Ah	**Line feed or cursor down.** Moves the cursor down one row without affecting the current column position.
0Bh	**Cursor up.** Moves the cursor up one row without affecting the current column position.
0Ch	**Cursor right.** Moves the cursor one column position to the right without altering the character under the cursor.

0Dh	**Carriage return**. Returns the cursor to the first column position of the current row.
11h	**Clear to the end of the screen**. Changes all characters to spaces beginning with the current cursor position to the end of the screen. The position of the cursor remains unchanged. Characters before the cursor remain unchanged.
18h	**Clear to the end of line**. Changes all characters from the current cursor position to the end of the current line to spaces. The cursor position is unchanged. Characters before the cursor are unchanged.
1Ah	**Clear screen and home cursor**. Clears the entire screen and places the cursor in the home position (column 0, row 0).
1Bh	**Escape**. The first character of an escape sequence. These sequences are explained on this page and the next.
1Eh	**Home Cursor**. Moves the cursor to the home position (column 0, row 0) without otherwise affecting the screen display.
1Fh	**Display next character direct**. After receiving this code, the next character is displayed directly on the screen without interpreting it as a special display function code. This code is usually used to display control characters that are not normally displayed by the ROSR.

Display Manipulation through ESC codes
Listed below is a summary table of the multi-character sequences used to manipulate the display. Each sequence's effect is more fully described in the text following the table. Note that all of these sequences are all preceded by the escape character 1Bh.

ESCAPE SEQUENCES

<ESC> followed by	Function
28h	Disable attribute display
29h	Enable attribute display
2Ah	Clear screen
30h	Pass 7-bit keyboard data
31h	Pass 8-bit keyboard data
34h	Set blink attribute mode
35h	Set graphics attribute mode
36h	Set blink attribute mode
37h	Set inverse video attribute mode
38h	Set low intensity attribute mode
3Dh	XY cursor position lead-in
45h	Line insert
51h	Character insert
52h	Line delete
57h	Character delete

28h **Disable attribute display**. Will cause all succeeding characters displayed on the screen to unconditionally have the upper bit reset, so that the selected attribute mode will not be displayed. Display will continue in this mode until changed by the <ESC> 29h sequence code.

29h **Enable attribute display**. Setting this mode will cause all following characters displayed on the screen to unconditionally have the upper bit set, thereby causing the selected attribute mode to be displayed. This mode will continue in effect until the <ESC> 28h code disables it.

2Ah	**Clear screen.** This function clears the screen to spaces with the cursor at the home position.
30h	**Pass only 7 bits of data from the keyboard.** This is the default setting at power-on (or reset), and is compatible with the 820. This mode of operation does not allow many of the unique codes generated by the keyboard to be used by applications software. A corollary effect is also automatically engaged in the 7-bit mode. Only 7 bits of data will be passed to the video display screen. ASCII characters with the upper bit set will normally cause one of the four attributes to be displayed (blink, lowlight, inverse video, or graphics characters). The <ESC> 30h code prevents this sometimes undesired feature.
31h	**Pass the upper bit of data from the keyboard.** Using the <CTRL> key along with certain keys will set the upper (eighth) bit of that key, allowing these codes to be processed as special function keys by applications programs. The following 30 keys produce unique codes.

Ctrl + Key	**Numeric Pad (hex)**	
0	B0	
1	B1	
2	B2	
3	B2	
4	B4	
5	B5	
6	B6	
7	B7	
8	B8	
9	B9	
period	AE	
plus sign	AB	
minus sign	AD	
up arrow	81	
down arrow	82	
right arrow	83	
left arrow	84	
line feed	8A	
DEL	FF	Reserved

Ctrl + Key	Main Keyboard (hex)	
1	91	
2	92	
3	93	
4	94	
5	95	
6	96	
7	97	
8	98	
9	99	
=	9A	
backspace	88	Reserved
tab	89	
return	8D	Reserved

34h or 36h **Set blinking attribute mode**. This code will not actually begin displaying blinking characters on the screen. **Note**: All the "set attribute mode" code sequences work in the same manner. An <ESC> 29h sequence is used to enable the display of the attribute characters, or storing characters on the screen with the upper bit set, as described above. Thus, any of the different attribute modes can be selected without affecting the screen display as long as there are NO characters on the screen with the upper bit set. If there ARE characters displayed on the screen with upper bit set, changing attribute modes will cause an IMMEDIATE change in the way the upper bit characters are displayed, depending on the attribute mode selected.

35h **Set graphic character attribute mode**. See **Note**: above.

37h **Set inverse video attribute mode**. See **Note**: above.

38h **Set low intensity attribute mode**. See **Note**: above. It should be noted that low intensity is the default attribute mode. The CP/M CONFIGUR program allows you to select your own default attribute mode.

3Dh **Position the cursor to the location indicated** by the following two row and column codes. The "home" position is designated as row 0, column 0. An offset of 20 hex must be added to the X and Y position codes. The positioning formula is:

ESC = (X + 20h) (Y + 20h)

where legal X (row) values are between 0 and 23 and legal Y (column) values are between 0 and 79.

45h **Line insert**. Will move the entire line on which the cursor resides down one line, filling the cursor line with spaces, and causing the line on the bottom of the screen to disappear. (It is actually moved to the internal command line buffer for the monitor so that applications programs wishing to preserve the bottom line are able to do so.) The actual position of the cursor will not change.

51h **Character insert**. Will insert a space at the current cursor position, causing the character under the cursor and all characters after the cursor to be shifted one position to the right. The last character on the line will disappear. The cursor position will remain unchanged and the character under the cursor will be the inserted space. No other lines will be affected. The character that was "lost" at the end of the line will actually be placed into the A register and the HL register will be pointing to the current cursor position upon return from the Fast CRT jump vector entry point (0F00Fh) so that applications programs can preserve this character.

52h **Line delete**. Similar to the line insert function except that the line on which the cursor resides will be deleted from the screen (and moved to the line buffer as described above), and all lines below it will be moved up one line. The position of the cursor will be unchanged.

57h **Character delete**. This function will delete the character under the cursor and cause all characters to the right of the cursor to move one position to the left. The last character position of the line will be replaced by a space. The cursor position will be unchanged and the character under the cursor will now be the character that was to the immediate right of the cursor before the character delete operation. The deleted character will be placed into the A register and the HL register will be pointing to the current cursor position upon return from the Fast CRT jump vector entry point (0F00Fh) so that applications programs can preserve this character.

54h	**Clear to End of Line**. Changes all characters from the current cursor position to the end of the current line to spaces. The cursor position is unchanged. Characters before the cursor position are unchanged.
59h	**Clear to End of Screen**. Changes all characters to spaces beginning with the current cursor position to the end of the screen. The position of the cursor remains unchanged. Characters before the cursor position remain unchanged.

Notes

ASCII Keyboard

Main Key Array:	50 keys plus 3 modifier keys (Alpha Lock, Shift, and Control)
Numeric Key Pad:	20 keys to the right of the main array
Cursor Keys:	4 keys on numeric key pad
Interface	U.S. ASCII-Coded Parallel Interface
Engraving:	U.S. Standard ASCII Keycaps

The electronic keyboard uses a standard 96-character ASCII keyboard. A ten-key numeric pad is included for typing statistical material. Parallel output is standard. A list of the output codes (in hex) for the Unshifted, Shifted, and CTRL + sequences for this keyboard begins on page 198.

Auto-Repeat

When an auto-repeat key is pressed, the following will be generated:

- internal code output
- pause of 0.5 (± 0.1) seconds
- repeat code output at rate of 16 (± 1) characters per second
- code output terminates immediately upon release of key

Repeat Keys	Keystation
- (minus)	12
= (equal)	13
backspace	14
delete	15
return	50
line feed	51
up arrow	52
x	58
. (period)	65
/ (slash)	66
left arrow	68
down arrow	69
right arrow	70
space bar	74

Function Key Priority

When more than one function key is pressed, the output will use the function key with the highest priority. The priority of the function keys in descending order is: Shift, CTRL, Lock, Unshifted.

Function Key Uses

The Shift function (keystations 56 and 57) causes production of shift-key-codes. Affected keystations are:

Key	Keystation	Key	Keystation
1	2	A	39
2	3	S	40
3	4	D	41
4	5	F	42
5	6	G	43
6	7	H	44
7	8	J	45
8	9	K	46
9	10	L	47
0	11	; (semi-colon)	48
- (minus)	12	' (apostrophe)	49
= (equal)	13		
		Z	57
Q	21	X	58
W	22	C	59
E	23	V	60
R	24	B	61
T	25	N	62
Y	26	M	63
U	27	, (comma)	64
I	28	. (period)	65
O	29	/ (slash)	66
P	30		
[	31		
]	32		

Alpha Lock

The Alpha Lock key (keystation 38) mechanically locks in the down position when first pressed and releases when pressed a second time. The Alpha Lock key activates the Shift key function for the keystations listed below.

Key	Keystation	Key	Keystation
Q	21	F	42
W	22	G	43
E	23	H	44
R	24	J	45
T	25	K	46
Y	26	L	47
U	27	Z	57
I	28	X	58
O	29	C	59
P	30	V	60
A	39	B	61
S	40	N	62
D	41	M	63

CTRL

The CTRL key (keystations 73 and 75) allows almost every key on the board to have a second or third output code. The CTRL key is used in conjunction with the alphabetic keys as function keys and to access the complete set of ASCII codes.

The chart beginning on the next page lists the output codes in hex for the Unshifted, Shifted, and CTRL + sequences for the ASCII keyboard. Bolding indicates a Reserved key.

Key	#	Unshifted	Shifted	CTRL +
Help	01	1E	1E	**9E**
1	02	31	21	91
2	03	32	40	92
3	04	33	23	93
4	05	34	24	94
5	06	35	25	95
6	07	36	5E	96
7	08	37	26	97
8	09	38	2A	98
9	10	39	28	99
0	11	30	29	90
- (Minus)	12	2D	5F	1F
= (Equal)	13	3D	2B	9A
Backspace	14	08	08	**88**
Del	15	7F	7F	**FF**
- (Minus - pad)	16	2D	2D	AD
7 (pad)	17	37	37	B7
8 (pad)	18	38	38	B8
9 (pad)	19	39	39	B9
Tab	20	09	09	89
q	21	71	51	11
w	22	77	57	17
e	23	65	45	05
r	24	72	52	12
t	25	74	54	14
y	26	79	59	19
u	27	75	55	15
i	28	69	49	09
o	29	6F	4F	0F
p	30	70	50	10
[	31	5B	7B	1B
]	32	5D	7D	1D
ESC	33	1B	1B	**9B**
+ (Plus - pad)	34	2B	2B	AB
4 (pad)	35	34	34	B4
5 (pad)	36	35	35	B5
6 (pad)	37	36	36	B6
Lock	38	---	---	---
a	39	61	41	01
s	40	73	53	13

Key	#	Unshifted	Shifted	CTRL +
d	41	64	44	04
f	42	66	46	06
g	43	67	47	07
h	44	68	48	08
j	45	6A	4A	0A
k	46	6B	4B	0B
l	47	6C	4C	0C
; (Semi Colon)	48	3B	3A	7E
' (Apostrophe)	49	27	22	60
Return	50	0D	0D	**8D**
Line Feed	51	0A	0A	8A
Up Arrow	52	81	81	01
1	53	31	31	B1
2	54	32	32	B2
3	55	33	33	B3
Left Shift	56	---	---	---
z	57	7A	5A	1A
x	58	78	58	18
c	59	63	43	03
v	60	76	56	16
b	61	62	42	02
n	62	6E	4E	0E
m	63	6D	4D	0D
. (Comma)	64	2C	3C	1C
. (Period)	65	2E	3E	7C
/ (Slash)	66	2F	3F	5C
Right Shift	67	---	---	---
Left Arrow	68	84	84	04
Down Arrow	69	82	82	02
Right Arrow	70	83	83	03
0	71	30	30	B0
. (Period - pad)	72	2E	2E	AE
Left CTRL	73	---	---	---
Space	74	20	20	00
Right CTRL	75	---	---	---

--- = Function key
Bolding = Reserved key

Low Profile Keyboard

Main Key Array:	50 keys plus 3 modifier keys (Alpha Lock, Shift, and Control)
Numeric Key Pad:	18 keys to the right of the main array
Function Keys	12 keys above main array key pad labelled F1 through F12
Cursor Keys:	5 keys to the right of the main array including Home key
Other Keys:	7 keys to the right of main array such as Help, Accept, Delete, etc.
Interface	U.S. ASCII-Coded Parallel Interface
Engraving:	U.S. Standard ASCII Keycaps

The electronic keyboard uses 97 key positions, follows the standard 96-character ASCII keyboard layout, and has one level of position-encoding. All keys generate their own unique position code on both the up and downstroke. Two bytes of data, command status and position, are output for each unique key motion. Parallel output is standard.

The standard output is two bytes of serial data. (A third byte will be used for mouse data.) Each byte is composed of a start bit, eight data bits, and a stop bit. The bytes are spaced 700 (± 150) microseconds apart. The first byte, or command status word, defines the status of the keyboard and the meaning of the data in the next byte. The second byte, or position word number one, defines the position of the key. If mouse data is present, this byte defines the X movement of the mouse. The third byte, or position word number two, defines the Y movement of the mouse. Note that the third byte is present only for mouse data.

Data Format

Command Status Word

Bit 0	Alpha Lock
Bit 1	Shift - Left or Right
Bit 2	Control - Left or Right
Bit 3	Mouse Data
Bit 4	-X if set; + X or 0 if not set
Bit 5	-Y if set; + Y or 0 if not set
Bit 6	Up/Downstroke (up = set)
Bit 7	Always Set (indicates command staus word)

Data Format continued

Position Word Number One

Bit 0-6	XY position or mouse travel (ΔX Magnitude)
Bit 7	Never Set - Indicates Data Word

Position Word Number Two (Only if mouse data)

Bit 0-6	Mouse Travel (ΔY Magnitude)
Bit 7	Never Set - Indicates Data Word

Keyboard Handler

The 16/8 operating system comes standard with the ASCII keyboard handler. This handler is not compatible with the optional position-encoded Low Profile Keyboard. The position-encoded keyboard handler and translation tables are located in the fourth ROM. The 16/8 system, when booting, identifies the position-encoded keyboard by checking the presence of the fourth ROM. If detected, a subroutine call is made to the fourth ROM to move the position-encoded keyboard handler and translation tables to RAM.

The position-encoded keyboard handler inputs the keyboard command bytes and key-station codes. The command bytes identify the required action to be taken by the handler, and the valid keystation codes are used to index into the translation tables to recover the translated hexadecimal codes. The position-encoded handler returns the translated code to the ASCII keyboard handler to queue.

The keyboard handler has three keyboard translation tables. Each table consisits of 102 decimal bytes. The tables are RAM resident, and define the unshifted, shifted, and control + sequence states. The output codes for these three states begin on the next page (bolding indicates a Reserved key). The position-encoded keyboard handler listing can be found in Appendix J.

The exception tables identify the repeat keys, inhibited keys, shift-lock status, additional alpha lock codes, and mouse status. The mouse interrupt handler translates the mouse movement into display coordinates. The repeat key interrupt handler provides the timing for the repeat key functions. The default keyboard table recovery restores the ROM keyboard tables to RAM. The ROM tables contain only the U.S. ASCII translation codes.

Key	#	Unshifted	Shifted	CTRL +
ESC	01	1B	1B	**9B**
1	02	31	21	91
2	03	32	40	92
3	04	33	23	93
4	05	34	24	94
5	06	35	25	95
6	07	36	5E	96
7	08	37	26	97
8	09	38	2A	98
9	10	39	28	99
0	11	30	29	90
- (Minus)	12	2D	5F	1F
= (Equal)	13	3D	2B	9A
Backspace	14	08	08	**88**
Tab	15	09	09	89
q	16	71	51	11
w	17	77	57	17
e	18	65	45	05
r	19	72	52	12
t	20	74	54	14
y	21	79	59	19
u	22	75	55	15
i	23	69	49	09
o	24	6F	4F	0F
p	25	70	50	10
[	26	5B	7B	1B
]	27	5D	7D	1D
Return	28	0D	0D	**8D**
Left CTRL	29	---	---	---
a	30	61	41	01
s	31	73	53	13
d	32	64	44	04
f	33	66	46	06
g	34	67	47	07
h	35	68	48	08
j	36	6A	4A	0A
k	37	6B	4B	0B
l	38	6C	4C	0C
; (Semi Colon)	39	3B	3A	7E

Key	#	Unshifted	Shifted	CTRL +
' (Apostrophe)	40	27	22	60
Line Feed	41	0A	0A	8A
Left Shift	42	---	---	---
. (Period 10-key)	43	2E	2E	AE
z	44	7A	5A	1A
x	45	78	58	18
c	46	63	43	03
v	47	76	56	16
b	48	62	42	02
n	49	6E	4E	0E
m	50	6D	4D	0D
. (Comma)	51	2C	3C	1C
. (Period)	52	2E	3E	7C
/ (Slash)	53	2F	3F	5C
Right Shift	54	---	---	---
Help	55	1E	1E	9E
Right CTRL	56	---	---	---
Space	57	20	20	00
Lock	58	---	---	---
F1	59	F1	F1	D1
F2	60	F2	F2	D2
F3	61	F3	F3	D3
F4	62	F4	F4	D4
F5	63	F5	F5	D5
F6	64	F6	F6	D6
F7	65	F7	F7	D7
F8	66	F8	F8	D8
F9	67	F9	F9	D9
F10	68	FA	FA	DA
F11	69	FB	FB	DB
F12	70	FC	FC	DC
7	71	37	37	B7
8	72	38	38	B8
9	73	39	39	B9
, (Comma)	74	2C	2C	AC
4	75	34	34	B4
5	76	35	35	B5

Key	#	Unshifted	Shifted	CTRL +
6	77	36	36	B6
Enter	78	0D	0D	3D
1	79	31	31	B1
2	80	32	32	B2
3	81	33	33	B3
0	82	30	30	B0
Next	83	E7	E7	C7
Down Arrow	84	82	82	02
Left Arrow	85	84	84	04
Right Arrow	86	83	83	03
Home	87	80	80	1E
Up Arrow	88	81	81	01
Prev	89	E6	E6	C6
Accept	90	FD	FD	DD
Del	91	7F	7F	**FF**
+ (Plus)	92	2B	2B	AB
- (Minus)	93	2D	2D	AD
x (Multiply)	94	2A	2A	AA
÷ (Divide)	95	2F	2F	AF
Blank Key	96	F0	F0	D0
Undo	97	18	18	DE
Mouse:				
Switch 1	98	8E	8E	8E
Switch 2	99	8F	8F	8F

--- = Function key
Bolding = Reserved key

Note: Approximately 1,000 of the first-issued Low Profile keyboards will generate a unique code for the Enter key (10-key pad). These keyboards can be identified by reading the ROM sign-on message when you turn on the display. If your monitor displays a (V13) message you have one of the first 1,000 units and the Enter key output codes are BDh, BDh, and FEh. With (V16) and higher ROM levels, the Enter Key generates the codes listed above.

Auto-Repeat

When an auto-repeat key is pressed, the following will be generated:

- internal code output
- pause of 0.5 (± 0.1) seconds
- repeat code output at rate of 16 (± 1) characters per second
- code output terminates immediately upon release of key

Repeat Keys	Keystation
- (minus)	12
= (equal)	13
backspace	14
return	28
line feed	41
x	45
. (period)	52
/ (slash)	53
space bar	57
down arrow	84
left arrow	85
right arrow	86
up arrow	88
delete	91

Function Key Uses

The Shift function (keystations 42 and 54) causes production of shift-key-codes. Since every keystation is position-encoded, almost all keys output a unique hex code. For a complete list, refer to the Low Profile Keyboard Keystation chart beginning on page 202.

The Alpha Lock key (keystation 58) locks in the down position when first pressed (the red led light comes on), and releases when pressed again. The Alpha Lock key activates the Shift key function for the keystations listed in the chart on the next page.

Key	Keystation	Key	Keystation
Q	16	F	33
W	17	G	34
E	18	H	35
R	19	J	36
T	20	K	37
Y	21	L	38
U	22	Z	44
I	23	X	45
O	24	C	46
P	25	V	47
A	30	B	48
S	31	N	49
D	32	M	50

The CTRL keys (keystations 73 and 75) allow almost every key on the board to have a second or third output code. The CTRL keys are used in conjunction with the alphabetic keys as function keys and to allow access to the complete set of ASCII codes.

SA400L (5¼" Single-sided floppy)

	Single Density	**Double Density**
Capacity		
Unformatted	125 k bytes	250 k bytes
Formatted	90 k bytes	168 k bytes
Usable	**81 k bytes**	**155 k bytes**
Transfer Rate	125 k bits/sec	250 k bits/sec
Latency (average)	100 Ms	100 Ms
Access Time		
Track to Track	20 Ms	20 Ms
Average	275 Ms	275 Ms
Settling Time	20 Ms	20 Ms
Rotational Speed	300 RPM	300 RPM
Recording Density	2768 BPI	5536 BPI
Flux Density	5536 FCI	5536 FCI
Track Density	48 TPI	48 TPI
Tracks	40	40
R/W Heads	1	1
Physical Sectors per track	18	17
Bytes per Sector	128	256
Encoding Method	FM	MFM

DC Voltage Requirements

+ 12 V dc ± 5% @1.80A Maximum, 0.9A Typical
+ 5 V dc ± 5% @0.70A Maximum, 0.5A Typical

Power Dissipation

13.3 Watts (45.3 BTU/Hr) Continuous (Typical)
7.3 Watts (24.9 BTU/Hr) Standby (Typical)

SA450 ($5\frac{1}{4}$" Double-sided floppy)

	Single Density	Double Density
Capacity		
Unformatted	250 k bytes	508 k bytes
Formatted	180 k bytes	338 k bytes
Usable	**172 k bytes**	**322 k bytes**
Transfer Rate	125 k bits/sec	250 k bits/sec
Latency (average)	100 Ms	100 Ms
Access Time		
Track to Track	20 Ms	20 Ms
Average	275 Ms	275 Ms
Settling Time	15 Ms	15 Ms
Rotational Speed	300 RPM	300 RPM
Recording Density	2938 BPI	5876 BPI
Flux Density	5876 FCI	5876 FCI
Track Density	48 TPI	48 TPI
Tracks	40	40
R/W Heads	2	2
Physical Sectors per track	18	17
Bytes per Sector	128	256
Encoding Method	FM	MFM

DC Voltage Requirements

+ 12 V dc ± 5% @1.80A Maximum
+ 5 V dc ± 5% @0.70A Maximum

Power Dissipation

11.5 Watts (40 BTU/Hr) Continuous (Typical)
7.3 Watts (25 BTU/Hr) Standby (Typical)

SA800 (8" Single-sided floppy)

	Single Density	Double Density
Capacity		
Unformatted	400 k bytes	800 k bytes
Formatted	250 k bytes	497 k bytes
Usable	**241 k bytes**	**482 k bytes**
Transfer Rate	250 k bits/sec	500 k bits/sec
Latency (average)	83 Ms	83 Ms
Access Time		
Track to Track	8 Ms	8 Ms
Average	210 Ms	210 Ms
Settling Time	8 Ms	8 Ms
Head Load Time	35 Ms	35 Ms
Rotational Speed	360 RPM	360 RPM
Recording Density (inside track)	3268 BPI	6536 BPI
Flux Density	6536 FCI	6536 FCI
Track Density	48 TPI	48 TPI
Tracks	77	77
R/W Heads	1	1
Physical Sectors per track	26	26
Bytes per Sector	128	256
Encoding Method	FM	MFM

AC Power Requirements
60 Hz ± 0.5 Hz
115 V ac 85 to 127 V @ 0.4 A typical

DC Voltage Requirements
+ 24 V dc ± 5% 1.3 A typical
+ 5 V dc ± 5% 0.8 A typical
- 5 V dc ± 5% 0.5 A typical

Heat Dissipation = 274 BTU/hr typical (80 watts)

SA850 (8" Double-sided floppy)

	Single Density	Double Density
Capacity		
Unformatted	800 k bytes	1.6 M bytes
Formatted	500 k bytes	1 M byte
Usable	**490 k bytes**	**980 k bytes**
Transfer Rate	250 k bits/sec	500 k bits/sec
Latency (average)	83 Ms	83 Ms
Access Time		
Track to Track	3 Ms	3 Ms
Average	91 Ms	91 Ms
Settling Time	15 Ms	15 Ms
Head Load Time	50 Ms	50 Ms
Rotational Speed	360 RPM	360 RPM
Recording Density (inside track)	3408 BPI	6816 BPI
Flux Density	6816 FCI	6816 FCI
Track Density	48 TPI	48 TPI
Cylinders	77	77
Tracks	154	154
R/W Heads	2	2
Physical Sectors per track	26	26
Bytes per Sector	128	256
Encoding Method	FM	MFM

AC Power Requirements
- 60 Hz ± 0.5 Hz
- 115 V ac 85 to 127 V @ 0.35 A Max

DC Voltage Requirements
- +24 V dc ± 10% 1.0 A Max
- +5 V dc ± 5% 1.1 A Max

Heat Dissipation = 200 BTU/hr typical (60 watts)

SA1004 (8" Rigid)

Capacity	
Unformatted	10.67 M bytes
Formatted	8.4 M bytes
Usable	**8.192 M bytes**
Transfer Rate	4.34 M bits/sec
Latency (average)	9.6 Ms
Access Time	
Track to Track	1 Ms
Average	52 Ms
Maximum	132 Ms
Settling Time	18 Ms
Rotational Speed	3125 ± 3% RPM
Recording Density	6270 BPI
Flux Density	6270 FCI
Track Density	172 TPI
Tracks	1024
Cylinders	256
R/W Heads	4
Physical Sectors per track	32
Bytes per Sector	256

AC Power Requirements

60 Hz ± 0.5 Hz

115 V ac 90 to 127 V @ 1.1 A typical

DC Voltage Requirements

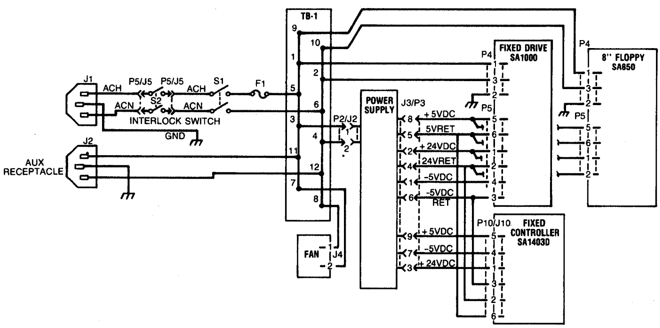
+ 24 V dc ± 2.4V 2.8A Typical 3.3A Maximum 1000mv ripple P-P

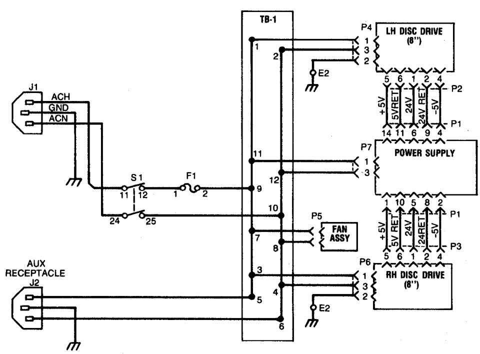
- 24 V dc ± 2.4V 2.8A Typical 3.3A Maximum 1000mv ripple P-P

+ 5 V dc ± 0.25V 2.0A Typical 2.5A Maximum 50mv ripple P-P

- 5 V dc ± 0.25V .20 A Typical .25A Maximum 50mv ripple P-P

Heat Dissipation = 321 BTU/hr typical (94 watts)





Floppy Disk Parameter Header (DPH)

When the physical disk driver is requested to identify the type of media presently installed in the disk drive, it returns the address of a CP/M 2.2-compatible disk parameter header in the HL register if the identification was successful. The first word (address) of this data structure is the address of the sector translate table. If the media is a single density disk there will be an address in the first field; if the media type is double density (or a rigid disk), the sector translate field of the disk paramater header will be 0000. This is because the sectors are physically skewed on the disk and the translate table is not necessary.

Single Density Logical/Physical Translate Tables

$5\frac{1}{4}$" Single density

1,6,11,16,3,8,13,18,5,10,15,2,7,12,17,4,9,14

8" Single density

1,7,13,19,25,5,11,17,23,3,9,15,21,2,8,14,20,26,6,12,18,24,4,10,16,22

The Disk Parameter Block (DPB)

The sixth address field in the disk parameter header (DPH) is the address for a disk parameter block to describe the physical disk. Listed below are the disk parameter blocks.

$5\frac{1}{4}$" Single-Sided Single Density

Sectors Per Track **(SPT)**	18
Block Shift Factor **(BSH)**	3
Max. Rec. # in blk **(BLM)**	7
Extent Mask **(EXM)**	0
Total Storage capacity **(DSM)**	82
Total # of directory entries **(DRM)**	31
Allocation mask **(AL0)**	80H
Allocation mask **(AL1)**	0
Size of dir check vector **(CKS)**	8
Reserved tracks **(OFF)**	3

$5\frac{1}{4}$" Single-Sided Double Density

Sectors Per Track **(SPT)**	34
Block Shift Factor **(BSH)**	3
Max. Rec. # in blk **(BLM)**	7
Extent Mask **(EXM)**	0
Total Storage capacity **(DSM)**	156
Total # of directory entries **(DRM)**	63
Allocation mask **(AL0)**	C0H
Allocation mask **(AL1)**	0
Size of dir check vector **(CKS)**	16
Reserved tracks **(OFF)**	3

5¼" Double-Sided Single Density

Sectors Per Track **(SPT)**	18
Block Shift Factor **(BSH)**	3
Max. Rec. # in blk **(BLM)**	7
Extent Mask **(EXM)**	0
Total Storage capacity **(DSM)**	172
Total # of directory entries **(DRM)**	31
Allocation mask **(AL0)**	80H
Allocation mask **(AL1)**	0
Size of dir check vector **(CKS)**	8
Reserved tracks **(OFF)**	3

5¼" Double-Sided Double Density

Sectors Per Track **(SPT)**	34
Block Shift Factor **(BSH)**	4
Max. Rec. # in blk **(BLM)**	15
Extent Mask **(EXM)**	1
Total Storage capacity **(DSM)**	162
Total # of directory entries **(DRM)**	63
Allocation mask **(AL0)**	C0H
Allocation mask **(AL1)**	0
Size of dir check vector **(CKS)**	16
Reserved tracks **(OFF)**	3

8" Single-Sided Single Density

Sectors Per Track **(SPT)**	26
Block Shift Factor **(BSH)**	3
Max. Rec. # in blk **(BLM)**	7
Extent Mask **(EXM)**	0
Total Storage capacity **(DSM)**	242
Total # of directory entries **(DRM)**	63
Allocation mask **(AL0)**	C0H
Allocation mask **(AL1)**	0
Size of dir check vector **(CKS)**	16
Reserved tracks **(OFF)**	2

8" Single-Sided Double Density

Sectors Per Track **(SPT)**	52
Block Shift Factor **(BSH)**	4
Max. Rec. # in blk **(BLM)**	15
Extent Mask **(EXM)**	1
Total Storage capacity **(DSM)**	242
Total # of directory entries **(DRM)**	127
Allocation mask **(AL0)**	C0H
Allocation mask **(AL1)**	0
Size of dir check vector **(CKS)**	32
Reserved tracks **(OFF)**	2

8" Double-Sided Single Density

Sectors Per Track **(SPT)**	26
Block Shift Factor **(BSH)**	4
Max. Rec. # in blk **(BLM)**	15
Extent Mask **(EXM)**	1
Total Storage capacity **(DSM)**	246
Total # of directory entries **(DRM)**	127
Allocation mask **(AL0)**	C0H
Allocation mask **(AL1)**	0
Size of dir check vector **(CKS)**	16
Reserved tracks **(OFF)**	2

8" Double-Sided Double Density

Sectors Per Track **(SPT)**	52
Block Shift Factor **(BSH)**	5
Max. Rec. # in blk **(BLM)**	31
Extent Mask **(EXM)**	3
Total Storage capacity **(DSM)**	246
Total # of directory entries **(DRM)**	127
Allocation mask **(AL0)**	C0H
Allocation mask **(AL1)**	0
Size of dir check vector **(CKS)**	32
Reserved tracks **(OFF)**	2

8" 8 Mb Rigid - Partition 1 - 4Mb

Sectors Per Track **(SPT)**	512
Block Shift Factor **(BSH)**	5
Max. Rec. # in blk **(BLM)**	31
Extent Mask **(EXM)**	1
Total Storage capacity **(DSM)**	3EFH
Total # of directory entries **(DRM)**	511
Allocation mask **(AL0)**	FFH
Allocation mask **(AL1)**	0
Size of dir check vector **(CKS)**	0
Reserved tracks **(OFF)**	1

8" 8 Mb Rigid - Partition 2 - 2Mb

Sectors Per Track **(SPT)**	512
Block Shift Factor **(BSH)**	5
Max. Rec. # in blk **(BLM)**	31
Extent Mask **(EXM)**	1
Total Storage capacity **(DSM)**	1EFH
Total # of directory entries **(DRM)**	511
Allocation mask **(AL0)**	FFH
Allocation mask **(AL1)**	0
Size of dir check vector **(CKS)**	0
Reserved tracks **(OFF)**	41H

8" 8 Mb Rigid - Partition 3 - 1Mb

Sectors Per Track **(SPT)**	512
Block Shift Factor **(BSH)**	5
Max. Rec. # in blk **(BLM)**	31
Extent Mask **(EXM)**	1
Total Storage capacity **(DSM)**	EFH
Total # of directory entries **(DRM)**	511
Allocation mask **(AL0)**	FFH
Allocation mask **(AL1)**	0
Size of dir check vector **(CKS)**	0
Reserved tracks **(OFF)**	61H

8" 8 Mb Rigid - Partition 4 - 1Mb

Sectors Per Track **(SPT)**	512
Block Shift Factor **(BSH)**	5
Max. Rec. # in blk **(BLM)**	31
Extent Mask **(EXM)**	1
Total Storage capacity **(DSM)**	EFH
Total # of directory entries **(DRM)**	511
Allocation mask **(AL0)**	FFH
Allocation mask **(AL1)**	0
Size of dir check vector **(CKS)**	0
Reserved tracks **(OFF)**	71H

Floppy Disk Physical Format

$5\frac{1}{4}$" Single Density floppy disk track format.

	Number of bytes	Hex Value of byte written
	22	FF
	4	00
	1	FE (I.D. Address Mark)
	1	Track Number
	1	Side Number (00 or 01)
	1	Sector Number
	1	00 (Sector length)
*	1	F7 (2 CRC's written)
	11	FF
	6	0
	1	FB (Data Address Mark)
	128	E5 (Data field)
	1	F7 (2 CRC's written)
	8	FF
	101	FF

* This field is repeated 18 times.

$5\frac{1}{4}$" Double Density floppy disk track format.

Number of bytes	Hex Value of byte written
50	4E
12	00
3	F5 (Writes A1)
1	FE (I.D. Address Mark)
1	Track Number
1	Side Number (00 or 01)
1	Sector Number
1	01 (Sector length)
1	F7 (2 CRC's written)
22	4E
12	0
1	FB (Data Address Mark)
256	E5 (Data field)
1	F7 (2 CRC's written)
32	4E
284	4E

* This field is repeated 17 times.

8" Single Density floppy disk track format.

	Number of bytes	**Hex Value of byte written**
	40	FF
	6	00
	1	FC (Index mark)
	26	FF
	6	00
	1	FE (I.D. Address Mark)
	1	Track Number
	1	Side Number (00 or 01)
	1	Sector Number
	1	00 (Sector length)
*	1	F7 (2 CRC's written)
	11	FF
	6	0
	1	FB (Data Address Mark)
	128	E5 (Data field)
	1	F7 (2 CRC's written)
	27	FF
	247	FF

* This field is repeated 26 times.

8 " Double Density floppy disk track format.

	Number of bytes	**Hex Value of byte written**
	80	4E
	12	00
	3	F6 (Writes C2)
	1	FC (Index mark)
	50	4E
	12	00
	3	F5 (Writes A1)
	1	FE (I.D. Address Mark)
	1	Track Number
	1	Side Number (00 or 01)
	1	Sector Number
*	1	01 (Sector length)
	1	F7 (2 CRC's written)
	22	4E
	12	0
	3	F5 (Writes A1)
	1	FB (Data Address Mark)
	256	E5 (Data field)
	1	F7 (2 CRC's written)
	54	4E
	247	FF

* This field is repeated 26 times.

Physical Disk Interleave

All double-density CP/M-formatted floppy disks have **Track 0, Side 0** formatted in single density. Also, double density floppies have the sectors physically skewed on the disk. This is done when the disk is formatted with the CP/M-80 program INIT.COM. Listed below are various physical sector placements for different system configurations and different options formatted with the INIT program. MS-DOS format information is listed on the next page.

5¼" Single Density
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18

5¼" Double Density
1, 7, 13, 2, 8, 14, 3, 9, 15, 4, 10, 16, 5, 11, 17, 6, 12

8" Single Density
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26

8" Double Density (820-II option with WD1797 disk controller)
1, 14, 8, 21, 2, 15, 9, 22, 3, 16, 10, 23, 4, 17, 11, 24, 5, 18, 12, 25, 6, 19, 13, 26, 7, 20

8" Double Density (16/8 option with WD1797 disk controller)
1, 22, 17, 12, 7, 2, 23, 18, 13, 8, 3, 24, 19, 14, 9, 4, 25, 20, 15, 10, 5, 26, 21, 16, 11, 6

8" Double Density (820-II option with SA1403D disk controller)
1, 10, 19, 2, 11, 20, 3, 12, 21, 4, 13, 22, 5, 14, 23, 6, 15, 24, 7, 16, 25, 8, 17, 26, 9, 18

8" Double Density (16/8 option with SA1403D disk controller)
1, 7, 13, 19, 25, 2, 8, 14, 20, 26, 3, 9, 15, 21, 4, 10, 16, 22, 5, 11, 17, 23, 6, 12, 18, 24

When the disk is formatted under the MS-DOS operating system, the gaps and skews apply to both single- and double-sided disks.

5¼" IBM PC Format
512 bytes per sector x 8 sectors per track

GAP1 = 50
GAP2 = 22
GAP3 = 96

Interleave = 1:1
(1, 2, 3, 4, 5, 6, 7, 8)

5¼" IBM XT Format
512 bytes per sector x 9 sectors per track

GAP1 = 50
GAP2 = 22
GAP3 = 96

Interleave = 1:1
(1, 2, 3, 4, 5, 6, 7, 8, 9)

8" Microsoft Format
256 bytes per sector x 26 sectors per track

GAP1 = 50
GAP2 = 22
GAP3 = 24

Interleave = 1:2
(1, 14, 2, 15, 3, 16, 4, 17, 5, 18, 6, 19, 7, 20, 8, 21, 9, 22, 10, 23, 11, 24, 12, 25, 13, 26)

Configuration Sector

The CONFIGUR.COM program writes the information listed below on track 0 of double density disks only.

8" floppy: Track 0, Sectors 2 and 3
$5\frac{1}{4}$" floppy: Track 0, Sectors 2 and 3
8" rigid: Track 0, Sector 20_{16}
8" rigid (2nd copy): Track 0, Sector 40_{16}

Byte offset

00-0e	Partition 1	(E:) DPB
0f	01H	(256 byte/sector)
10-1e	Partition 2	(F:) DPB
1f	01H	(256 byte/sector)
20-2e	Partition 3	(G:) DPB
2f	01H	(256 byte/sector)
30-3e	Partition 3	(G:) DPB
3f	01H	(256 byte/sector)
40-de		(Reserved for Rank Xerox language use)
df	Screen Attributes:	35h = Graphics, 36h = Blink, 37h = Inverse, 38h = Hi/lo Light
e0	Disk Head Step Rate:	0 = 3 msec, 1 = 6 msec, 2 = 10 msec, 3 = 15 msec (msec x2 if $5\frac{1}{4}$")
e1	83h	
e2	Keyboard bit mask	30h = 7-bit, 31h = 8-bit
e3-e7	Comm PIO command	08 06 00 18 03
e8	Comm Rec. Wd Length	41h = 7-bit, c1h = 8-bit
e9	04	

ea	Stop Bit & Parity	0100 xx yy	xx	= 01	1 stop bit
				= 11	2 stop bits
			yy	= 00	no parity
				= 01	odd parity
				= 10	no parity
				= 11	even parity
eb	05				
ec	Comm Transmit	1xx01010	xx	= 01	7 bit
		↑ ↑		= 11	8 bit
		DTR CTS			
ed-f1	Printer PIO Table	08 07 00 18 03			
f2	Receive	41h = 7 bit, c1h = 8 bits/character			
f3	04				
f4	Stop Bit & Parity	0100 xx yy	xx	= 01	1 stop bit
				= 11	2 stop bits
			yy	= 00	no parity
				= 01	odd parity
				= 10	no parity
				= 11	even parity
f5	05				
f6	Transmit	1 xx 01010	xx	= 01	7 bit
				= 11	8 bit
f7	Clear-to-Send	00 = Ignore, 01 = Low, 10 = High			
f8	80				
f9	Carrier Detect	00 = Ignore, 01 = Low, 10 = High			
fa	81				
fb	Protocol	ca = none, 00 = Xon/Xoff			
fc	82				
fd	Comm Baud	0xh (x = baud rate, see Appendix K for complete list)			
fe	Printer Baud	0xh (x = baud rate, see Appendix K for complete list)			
ff	IOBYTE (Z80-A)	xx 0000 yy			

	Default	00	01	10	11
xx = List Device	LPT	----	CRT	LPT	----
yy = Console	CRT	TTY	CRT	----	----

20 CPS Printer

The 20 CPS Printer (Diablo 620) is a serial printer designed for low-speed, low-to-moderate output requirements of standalone word and data processing business systems. The printer uses conventional data interchange techniques and protocol at speeds up to 1200 Baud.

This printer produces a "typewriter" quality output of fully-formed characters at a maximum of 20 Characters Per Second when printing average English (Shannon) text at 10 pitch. It includes operating features such as page formatting, graphics, positive and negative full and half line feed, absolute horizontal and vertical tabbing, as well as 12 and 15 pitch and proportional spacing.

The 20 CPS printer features a new 98-character plastic printwheel with automatic recognition of printwheel type and language. A "drop-in" printwheel exchange system is also featured (printwheels may be exchanged without removing the ribbon cartridge). The 20 CPS printer features quick-change film ribbon cartridges and has printwheels available in many languages and type styles.

Set the Switches Under the Front Cover

Check to be sure that the operating mode switches have been properly set for use. These switches set the printer parameters when printer power is turned on. If the switch settings are changed with the printer turned on, the changes will not have any effect until the power to the printer is turned off and then on again.

The switches are located to the left of the external control panel and are covered when the front cover is in place. These switches control operating modes and ordinarily do not require attention once they have been set.

When using a 20 CPS Printer with your personal computer, all the operating mode switches should be positioned to the right (off) except for switch #8. It should be positioned to the left (on).

Note: When using single strike ribbons, the 20 CPS Printer single strike mode must be enabled. If single strike mode is not enabled, the print quality will be unacceptable. See the Xerox 620 Printer (20 CPS) 630 Printer (40 CPS) Operators Guide, Order #610P72115, page 13, for instructions on using single strike ribbons.

Using the Control Panel Switches

These four switches are located to the right of the Control Panel where they are accessible to the operator with all covers on the machine. These are momentary-action switches activated by a touch of the finger.

RESET: This switch will clear an "error" indication and return the printer to operation. It will also return the printer to operation following a PAUSE command.

PAUSE: Touching this switch will cause the printer to stop printing without any loss of data, and the power indicator will go out. Printing is continued by pressing the RESET switch.

LINE FEED: Touching this switch initiates a single line feed. Action is repeated if the switch is held activated longer than 1/2 second. A line feed code will not, however, be transmitted.

FORM FEED: Touching this switch initiates a form feed to the next top-of-form position. A form feed code is not, however, transmitted.

The POWER Indicator

The power indicator glows whenever power is turned on to the printer and will flash for the following conditions:

- A parity error was detected with the PARITY switch on.
- The printer buffer (memory) has overflowed.
- The printer didn't receive a "Data Set Ready" signal.

Operating Mode Switches
The following information is a brief explanation for each of the 20 CPS Printer operating mode switches.

SWITCH	EXPLANATION
1	110-300 BAUD: This switch selects 110 or 300 Baud as the speed at which the printer will receive and transmit data. If 1200 Baud is selected (#8 1200/OPT switch ON), switch #1 doesn't have any effect.
2	PARITY ODD-EVEN: This is used in conjunction with Parity ON-OFF to determine the nature of parity information handling.
3	PARITY ON-OFF: This switch enables parity checking and parity information transmission when on.
4	DC1/DC3: This switch is used to allow the printer to operate with much faster host systems without loss of data. When ON, special characters (DC1/DC3) are transmitted between the printer and the host automatically whenever the print buffer (512 bytes) is either nearly empty (DC1) or nearly full (DC3).
5	SELF TEST: If this switch is in the ON position when the printer is turned on, the printer will enter a self test mode and begin sequencing through its self test program. The Control Panel PAUSE and RESET switches may be used to interrupt the self test sequence. To exit the mode, the SELF TEST switch must be moved to OFF and the power to the 20 CPS must be turned off momentarily.
6	AUTO/LF: When ON, this switch enables the printer to automatically advance the paper one line with each carriage return. This relieves the host system of the need to send a line feed command with each carriage return command.

SWITCH	EXPLANATION
7	PAGE SIZE: This switch enables setting page size, used in the Top Of Form/Form Feed function, to either the US standard 11" or the European standard 12" page length.
8	1200/OPT: This switch, when ON, enables the printer to receive and transmit data at a speed of 1200 Baud.

Horizontal Motion Index (HMI)
/Vertical Motion Index (VMI)

Horizontal Motion Indexing refers to the horizontal distance that the carriage moves for each character (or space) print command. Each increment is 1/120" and standard HMI values for 10, 12, and 15 pitch are:

10-Pitch = 12/120"   12-Pitch = 10/120"   15-Pitch = 8/120"

Vertical Motion Indexing (VMI) refers to the vertical distance that the paper moves for each line feed command. Each increment is 1/48".

Proportional Space provides its own HMI on a character to character basis as shown in the following example:

HMI = ESC US (n)

VMI = ESC RS (n)

In the example above, (n) is the decimal value of an ASCII character selected from the chart to produce the desired index value of (n-1). The minimum index value is 0 and the maximum index value is 125 in either the horizontal or vertical direction.

20 CPS Printer Operating Codes

The 20 CPS Printer uses two types of code to control the exchange and storage of information and the format of the printout. They are escape (ESC) codes, and control (CTRL) codes.

The ESC code (character) is used to pre-condition the printer logic to recognize the characters following the ESC code and preceding a carriage return as a command rather than print data.

20 CPS Printer Command Codes

MARGINS AND FORMATTING			
Set Top Page Margin (at current position)**	ESC	T	(n)
Set Left Margin (at current position)**	ESC	9	
Set Right Margin (at current position)**	ESC	0	
Set Lines Per Page to (n)*	ESC	**FF**	(n)
Set Bottom Page Margin (at current position)**	ESC	L	
Clear Top and Bottom Margins	ESC	C	
Set Horizontal Motion Index to (n - 1)*	ESC	**US**	(n)
Set Vertical Motion Index to (n - 1)*	ESC	**RS**	(n)
Return HMI control to internal program	ESC	S	
CARRIAGE MOVEMENT			
Absolute HT to print column (n)*	ESC	**HT**	(n)
Enable Auto Backward Printing	ESC	/	
Disable Auto Backward Printing	ESC	\	
Forward Print Mode ON	ESC	5	
Backward Print Mode ON (Forward Print Mode OFF) (clear with <return>)	ESC	6	
PAPER MOVEMENT			
Absolute VT to line (n)*	ESC	**VT**	(n)
Perform Negative Line Feed	ESC	**LF**	
Perform Half-Line Feed	ESC	U	
Perform Negative Half-Line Feed	ESC	D	

Note: the bolded characters indicate a control code. See ASCII Chart, page 234.

* (n) represents the decimal value of an ASCII character.

** The Left and Right Margin positions must be set by using SPACE and BACKSPACE commands from the Carriage Home (RESET) position. The Top and Bottom Margin positions must be set by using Line Feed commands or <return> from the manually-set Top of Form postion.

20 CPS Printer Command Codes continued

PRINTING			
Graphics Mode ON (clear w/<return>)	ESC	3	
Graphics Mode OFF	ESC	4	
WORD PROCESSING COMMANDS			
Proportional Space ON (clear with <return>)	ESC	P	
Proportional Space OFF	ESC	Q	
Program Mode ON (clear with ESC X or SI)	ESC	**SO**	M
Program Mode OFF	ESC	X	
MISCELLANEOUS COMMANDS			
Initiate Remote RESET	ESC	<ret>	P
Print, Printwheel Character 20h	ESC	Y	
Print, Printwheel Character 7Fh	ESC	Z	
Print, Printwheel Character 80h	ESC	a	
Print, Printwheel Character 81h	ESC	b	
Print, Printwheel Character 82h	ESC	c	
Print, Printwheel Character 83h	ESC	d	
Enable Download of Printwheel Conversion Table	ESC	**SO**	**DC2**
Enter Remote Diagnostic Mode (see next page also)	ESC	**SUB** (enter option)	

When selecting the remote diagnostic mode, enter one of the following options:

Remote Initialization	I
Remote Error Reset	R
STATUS 1 Request	1
RAM/ROM TEST	**SO**
Enter TEST MODE	**ENQ**

Note: the bolded characters indicate a control code. See ASCII Chart, page 234.

Remote Diagnostics

Status 1 "Word"	RAM/ROM Test "Word"	Test Mode "Word"
0 - (unassigned)	0 - 8041 RAM is bad	@, data byte - 8041 RAM data *
1 - 10 Pitch	1 - 8041 ROM is bad	A - Perform RAM/ROM check
2 - (unassigned)	2 - 6803 RAM is bad	B - Print 1 line swirltest
3 - (unassigned)	3 - 6803 ROM is bad (upper half 4K)	C - Print swirltest continuously
4 - (unassigned)	4 - 6803 ROM is bad (lower half 4K)	D - Stop printing swirltest
5 - Printer idle**	5 - (unassigned)	$40, data byte- 6803 RAM data*
6 - (unassigned)	6 - (unassigned)	DEL - Exit Test Mode
7 - UART Parity Bit	7 - UART Parity Bit	

* The Data Byte defines the RAM data address in ASCII code. The response is two bytes: 1) STX, ($02); and 2) contents of the RAM address requested.

**Print buffer empty and all printer motion complete.

COMMAND	CONTROL CODE	EQUIV. HEX CODE
ACK	CTRL F	06
BEL	CTRL G	07
BS	CTRL BACKSPACE or H	08
CAN	CTRL X	18
CR	CTRL <return> or M	0D
DC1	CTRL Q	11
DC3	CTRL S	13
DC4	CTRL T	14
DEL	CTRL DEL	7F
DLE	CTRL P	10
EM	CTRL Y	19
ENQ	CTRL E	05
EOT	CTRL D	04
ESC	CTRL ½ or[	1B
ETB	CTRL W	17
ETX	CTRL C	03
FF	CTRL L	0C
FS	CTRL \	1C
GS	CTRL è	1D
HT	CTRL TAB or I	09
LF	CTRL LF or J	0A
NAK	CTRL U	15
NUL	CTRL 1-8	00
RS	CTRL =	1E
SI	CTRL O	0F
SO	CTRL N	0E
SOH	CTRL A	02
SP	CTRL <space>	20
STX	CTRL B	02
SUB	CTRL Z	1A
SYN	CTRL V	16
US	CTRL -	1F
VT	CTRL K	0B
{	CTRL 9	7B
}	CTRL 0	7D
é	CTRL é	2C
.	CTRL .	2E
;	CTRL ;	3B
/	CTRL /	2F
'	CTRL '	27

Specifications

Print Speed:	Up to 20 characters per second printing average English (Shannon) text at 10 pitch.
Character Set:	98 character spaces consisting of 82 standard or common character segments. The printer supports several English and several foreign language printwheels.
Printwheels:	Plastic 98 character ASCII Xerox. The printer will automatically recognize printwheel pitch and language requirements.
Character Spacing:	10-pitch = 10 characters/inch (3.94 ch/cm) 12-pitch = 12 characters/inch (4.72 ch/cm) 15-pitch = 15 characters/inch (5.91 ch/cm)
Column Spacing:	1/120 inch (.21mm) minimum.
Print Line:	13.2 inches (335.3mm) 132 columns 10 pitch 158 columns 12 pitch 197 columns 15 pitch
Print Buffer:	512 bytes.
Paper Width:	13.2 inches (387.4mm) maximum - friction feed platen.
Carriage Speed:	1.7 seconds maximum for 13.2 inches (332.77mm) of motion.
Tabulation:	Left or right.
Line Spacing:	1/48 inch (.53mm) minimum.
Paper Feed:	Bidirectional.
Paper Thickness:	1 to 5 part forms; maximum overall thickness .024" (.61mm).
Sensors:	End of ribbon, paper out, and cover open.
Other Features:	Self test; host program control through escape sequences; data receive/transmit speed selection.

Power Requirements:	Operation from nominal 120/220-240 volt AC inputs, 50-60 Hz. 120W maximum power consumption. Check your printer's serial plate for proper input power.

Cabling Requirements

A standard RS-232-C interface cable is required for connection between the screen and the printer. This cable must be equipped with DB-25P connectors with the following pins connected:

PIN NO.	CCITT DESIG.	TELCO DESIG.	DESCRIPTION
1	101	AA	Protective Ground
2	103	BA	Transmitted Data
3	104	BB	Received Data
4	105	CA	Request To Send
6	107	CC	Data Set Ready *
7	102	AB	Signal Ground
20	108	CD	Data terminal Ready

* Pin 6 must be HI to receive or transmit data.

40 CPS Printer

The 40 CPS Printer (Diablo 630) is a medium-speed, daisy wheel serial printer. The 40 CPS printer is capable of producing typewriter quality output at speeds up to 40 characters per second with 88, 92, or 96 character metal printwheels (or 96 character plastic printwheel). The version sold by Xerox Corporation for use with the 820-II or 16/8 is a Model 630R132 which has the HPR05 PWA interface.

40 CPS Printer Versions

The 40 CPS Printer has three versions: Basic, Expanded, and Word Processor. The feature differences among the three versions of the HPR05 terminal are primarily a function of the firmware installed on the HPR05 circuit board in the form of the programmed ROM (Read Only Memory) devices and a nonvolatile RAM (Random Access Memory). The version offered by Xerox for use with the 820-II and 16/8 product uses the basic configuration of the HPR05 since printer control is taken by the 820-II and 16/8 system software for such applications as Word Processing.

HPR05 Communications Protocol

DC1/DC3 (XON/XOFF) protocol transmit a DC3 control code character from the 40 CPS Printer under any of the following conditions:

- Print buffer (2688 bytes) nearly full (within 64 bytes)
- Cover Open
- Paper Out (only when printing is attempted)
- End of Ribbon (only when printing is attempted)
- Printer in CHECK condition
- PAUSE switch depressed

A NAK character will be transmitted (in addition to the DC3) for: Cover Open, Paper Out, End of Ribbon, and CHECK condition if the HPR05 firmware is level -03 or later, and if both DC1/DC3 and EXT/ACK are enabled. The NAK character thus distinguishes the "error" condition from Buffer Full and PAUSE. NAK is also sent when a parity error is detected if parity checking is enabled. The error condition with the NAK can be cleared by pressing RESET.

The BUFFER FULL DC3 control character when transmitted by the 40 CPS Printer will be followed by a DC1 control character when the printer buffer (2688 bytes) is nearly empty (within 64 characters).

Setting the Switches Under the Access Cover
Check to be sure the printer has been set to the proper switch positions for use with a Xerox Personal Computer.

- Printwheel Select Switch. Set this switch to match the particular type of printwheel being used. This ensures your text will print correctly and prevents possible printwheel damage or excessive wear.

Check your printwheel to determine if it's plastic or metal and which pitch it is. The available printwheel settings are:

0: 88 Metal
2: 92 Metal
3: 96 Metal
4: 96D Metal
5: APL Metal
6: APL Plastic
7: Plastic (Normally shipped with the printer)
1,8,9: Optional

- Spacing Select Switch. This switch selects the horizontal spacing for character printout. Set this switch to 1 for 10 Pitch or 2 for 12 Pitch.

The available spacing settings are:

0: Proportional
1: 10 (Normally shipped with the printer)
2: 12
3: 15
4 - 9: Self Test

- Operating Mode Switches. When connecting the printer to a Xerox Personal Computer, the switches to the right of the Printwheel and Spacing switches should be positioned toward the front of the printer, except for the BAUD switch marked 120. It should be positioned toward the back of the printer.

The Power Indicator

The power Indicator should glow; the carriage should move to the left slowly, and then back to the right, to stop at the first print position; the printwheel should rotate and stop at its "home" position (i.e., the "flag" on metal printwheels should be at the top if the Printwheel Select switch - under the access cover - has been properly set). This entire process is called the INITIALIZATION, RESET, or RESTORE sequence. It clears all volatile memory, resets all position counters, and sets the printer to print the first character.

Using The Operating Switches

These six switches are located in the right-hand area of the control panel where they are accessible to the operator with all covers on the machine. These are membrane-type, momentary-action switches activated by a touch of the finger.

RESET Switch - This switch will restore the printer to normal operating status following a printer check or an error condition, and clears all error indicators.

SCROLL Switch - Touching this switch advances the paper a small amount to give the operator a clear view of the last printed line. The paper is automatically returned to the last printing position when the switch is released.

LINE FEED Switch - Touching this switch initiates a single or a double line feed operation, as selected by the DOUBLE L.F. MODE SWITCH. Action is repeated if the switch is held activated longer than 600 msec. A line feed code is not transmitted.

FORM FEED Switch - Touching this switch initiates a form feed to the next top-of-form position. A form feed code is not transmitted.

HERE IS Switch - Touching this switch causes a special "Here Is . ." message of up to 31 characters to be transmitted over the communications link when operating in remote ASCII mode with the fully featured HPRO5 option installed. This is not used with the 820, 820-II, or the 16/8.

BREAK Switch - Touching this switch causes a break (250 msec space) to be transmitted over the communications link when operating in remote mode.

Reading The Control Panel Indicators

POWER - Indicates that AC power is applied to the 40 CPS Printer.

PRINT CHK* - Indicates that a print operation has been called for while the printer is in a "check" condition. A check condition occurs when a printwheel or carriage movement command has been received but cannot be successfully completed due to a malfunction. This condition disables the printer until a restore sequence clears the check condition.

RESET - Note that if the problem causing the check condition has not been corrected when a restore sequence has been initiated, the check will reappear as soon as printing is attempted.

PARITY - This indicator functions only if the PARITY ENABLE switch (under the access cover) is ON. It indicates detection of any of the following types of errors:

- Incorrect parity sensed on received character.
- A framing error (no stop bit) detected on a received non-break character.
- A serial data character detected with an excess number of bits.

When a parity error is detected, a DEL character is substituted for the erroneous character.

OVERFLOW* - Indicates that the printer's print input memory (buffer) is too full (has overflowed). Protocol has not been used properly.

RIBBON/PAPER* - Indicates end of ribbon has been reached or that the printer is out of paper, and printing has been attempted.

COVER* - Indicates that printing was attempted with the cover open.

* These errors cause a break to be transmitted when the 40 CPS Printer is in Remote mode if DC1/DC3 protocol has not been selected.

Horizontal Motion Index (HMI) /Vertical Motion Index (VMI)

Horizontal Motion Indexing refers to the horizontal distance that the carriage moves for each character (or space) print command. Each increment is 1/120" and standard HMI values for 10, 12, and 15 pitch are:

10-Pitch = 12/120"   12-Pitch = 10/120"   15-Pitch = 8/120"

Vertical Motion Indexing (VMI) refers to the vertical distance that the paper moves for each line feed command. Each increment is 1/48".

Proportional Space provides its own HMI on a character to character basis as shown in the following example:

HMI = ESC US (n)

VMI = ESC RS (n)

In the example above, (n) is the decimal value of an ASCII character selected from the chart to produce the desired index value of (n-1). The minimum index value is 0 and the maximum index value is 125 in either the horizontal or vertical direction.

40 CPS Printer Operating Codes

The 40 CPS Printer uses two types of codes to control the exchange and storage of information and the format of the printout. They are escape (ESC) codes, and control (CTRL) codes.

The ESC code (character) is used to pre-condition the printer logic to recognize the characters following the ESC code and preceding a carriage return as a command rather than print data.

The CTRL key is used simultaneously with another key to generate an ASCII control signal to be used either internally or transmitted to the receiving system.

40 CPS Printer Command Codes

MARGINS AND FORMATTING			
Set Top Page Margin (at current position)**	ESC	T	
Set Left Margin (at current position)**	ESC	9	
Set Horizontal Tab Stop (HT) (at current position)**	ESC	1	
Set Right Margin (at current position)**	ESC	0	
Set Vertical Tab Stop (VT) (at current position)**	ESC	-	
Set Lines Per Page to (n)*	ESC	**FF**	(n)
Set Bottom Page Margin (at current position)**	ESC	L	
Clear Top and Bottom Margins	ESC	C	
Clear Horizontal Tab Stop (HT) (at current position)**	ESC	8	
Clear all HT and VT stops	ESC	2	
Set Horizontal Motion Index to (n - 1)*	ESC	**US**	(n)
Set Vertical Motion Index to (n - 1)*	ESC	**RS**	(n)
Return HMI control to internal program	ESC	S	
CARRIAGE MOVEMENT			
Absolute HT to print column (n)*	ESC	**HT**	(n)
Enable Auto Backward Printing	ESC	/	
Disable Auto Backward Printing	ESC	\	
Reverse Printing Mode	ESC	<	
Normal Printing Mode	ESC	>	
Forward Print Mode ON	ESC	5	
Backward Print Mode ON (Forward Mode OFF) (clear with <return>)	ESC	6	

Note: the bolded characters indicate a control code. See page 245.

* (n) represents the decimal value of an ASCII character.

** The Left and Right Margin positions must be set by using SPACE and BACKSPACE commands from the Carriage Home (RESET) position. The Top and Bottom Margin positions must be set by using Line Feed commands from the manually-set Top Of Form position.

40 CPS Printer Command Codes continued

PAPER MOVEMENT			
Absolute VT to line (n)*	ESC	**VT**	(n)
Perform Negative Line Feed	ESC	**LF**	
Perform Half-Line Feed	ESC	U	
Perform Negative Half-Line Feed	ESC	D	
PRINTING			
Graphics Mode ON (clear with <return>)	ESC	3	
Graphics Mode OFF	ESC	4	
WORD PROCESSING COMMANDS			
Proportional Space ON (clear with ESC S)	ESC	P	
Proportional Space OFF	ESC	Q	
Auto Underscore ON	ESC	E	
Auto Underscore OFF	ESC	R	
Bold Print ON (clear with <return>)	ESC	O	
Shadow Print ON (clear with <return>)	ESC	W	
Bold/Shadow Print OFF	ESC	&	
Backspace 1/120"	ESC	**BS**	
Program Mode ON (clear with SI)	ESC	**SO**	M
Program Mode OFF	ESC	X	
MISCELLANEOUS COMMANDS			
Initiate Remote RESET	ESC <ret>		P
Print, Printwheel Character 20h	ESC	Y	
Print, Printwheel Character 7Fh	ESC	Z	
Print, Printwheel Character 80h	ESC	a	
Print, Printwheel Character 81h	ESC	b	
Print, Printwheel Character 82h	ESC	c	
Print, Printwheel Character 83h	ESC	d	
Enable Download of Printwheel Conversion Table	ESC	**SO**	**DC2**
Enter Remote Diagnostic Mode (see next page)	ESC	**SUB**	(enter option)

Note: the bolded characters indicate a control code. See page 245.

When selecting the remote diagnostic mode, enter one of the following options:

Remote Initialization	I
Remote Error Test	R
Remote STATUS 1 Request	1
Remote RAM/ROM TEST	**SO**
Remote TEST MODE	**ENQ**

Note: the bolded characters indicate a control code. See page 245.

Status 1 "Word"	RAM/ROM Test "Word"	Test Mode "Word"
0 - (unassigned)	0 - 8041 RAM is bad	@, data byte - 8041 RAM data *
1 - 10 Pitch	1 - 8041 ROM is bad	A - Perform RAM/ROM check
2 - (unassigned)	2 - 6803 RAM is bad	B - Print 1 line swirltest
3 - (unassigned)	3 - 6803 ROM is bad (upper half 4K)	C - Print swirltest continuously
4 - (unassigned)	4 - 6803 ROM is bad (lower half 4K)	D - Stop printing swirltest
5 - Printer idle**	5 - (unassigned)	$40, data byte- 6803 RAM data*
6 - (unassigned)	6 - (unassigned)	DEL - Exit Test Mode
7 - UART Parity Bit	7 - UART Parity Bit	

* The Data Byte defines the RAM data address in ASCII code. The response is two bytes: 1) STX, ($02); and 2) contents of the RAM address requested.

**Print buffer empty and all printer motion complete.

COMMAND	CONTROL CODE		EQUIV. HEX CODE
ACK	CTRL	F	06
BEL	CTRL	G	07
BS	CTRL	BACKSPACE or H	08
CAN	CTRL	X	18
CR	CTRL	<return> or M	0D
DC1	CTRL	Q	11
DC3	CTRL	S	13
DC4	CTRL	T	14
DEL	CTRL	DEL	7F
DLE	CTRL	P	10
EM	CTRL	Y	19
ENQ	CTRL	E	05
EOT	CTRL	D	04
ESC	CTRL	½ or[	1B
ETB	CTRL	W	17
ETX	CTRL	C	03
FF	CTRL	L	0C
FS	CTRL	\	1C
GS	CTRL	è	1D
HT	CTRL	TAB or I	09
LF	CTRL	LF or J	0A
NAK	CTRL	U	15
NUL	CTRL	1-8	00
RS	CTRL	=	1E
SI	CTRL	O	0F
SO	CTRL	N	0E
SOH	CTRL	A	02
SP	CTRL	<space>	20
STX	CTRL	B	02
SUB	CTRL	Z	1A
SYN	CTRL	V	16
US	CTRL	-	1F
VT	CTRL	K	0B
{	CTRL	9	7B
}	CTRL	0	7D
é	CTRL	é	2C
.	CTRL	.	2E
;	CTRL	;	3B
/	CTRL	/	2F
'	CTRL	'	27

Electrical Interface Xerox 40 CPS Printer

EIA Interface Connector Pin Assignments

Pin #	Signal	Meaning
1	Chassis Ground	Connects to chassis ground within the 40 CPS printer.
2	-Transmitted Data	This connector is the serial ASCII-coded digital data being transmitted by the 40 CPS printer. This signal is in the "mark" state (LOW) between characters, rises for logic 0 and drops for logic 1.
3	-Received Data	This connector is the serial ASCII-coded digital data being received by the 40 CPS printer. This signal must be held in "mark" state (LOW) between characters. It should go HIGH for logic 0, and LOW for logic 1.
4	+ Request to Send	Held HIGH (+ 12VDC) whenever power is ON.
5	+ Clear to Send	(unused)
6	+ Data Set Ready	This connector must be ON (HIGH) for 40 CPS printer operation in Remote Mode. If OFF (LOW), no data can be received.
7	Signal to Ground	Ground reference for all interface signals.
8	+ Carrier Detect	The ON state of this signal is presented to the 40 CPS printer when the data communication equipment (DCE) is receiving a carrier signal suitable for demodulation. The OFF state indicates that no signal is being received by the DCE, or that the received signal is unsuitable for demodulation. In its present design, the 40 CPS printer ignores the Carrier Detect input signal.
11	+ Printer Ready	Goes LOW if any of the following conditions occur: • Print Buffer (2688 bytes) nearly full (within 64 bytes) • Cover Open • Paper Out • End of Ribbon • Printer in CHECK • Pause switch depressed

Pin #	Signal	Meaning
		With Paper Out or End of Ribbon, + Printer Ready goes LOW only if printing is attempted. It returns HIGH when the buffer becomes nearly empty, and/or conditions have been corrected.
20	+ Data Terminal Ready	ON (HIGH) whenever power is ON.

HPR05 Circuit Board Jumpers

Dipswitch Module A

#	Function		Meaning
1	Double Line Feed	ON-	Gives double line feed on every line feed command, and on every carriage return if switch 3 is ON.
		OFF-	Gives single line feed on every line feed command, and on every carriage return if switch 3 is ON.
2	(Unused)		
3	Auto Line Feed	ON-	Gives automatic line feed (single or double on every carriage return.
		OFF-	No line feed on carriage return. Line feed occurs only on separate line feed command.
4	(Unused)		
5	Uppercase Only	ON-	Converts all lowercase alpha characters (a-z) entered from the keyboard to uppercase.
		OFF-	Both uppercase and lowercase character selection, through the use of the shift key.
6	(Unused)		
7	Message Load	ON-	Enables keyboard entry of "Here Is..." message into non- volatile memory. (Functional only on expanded printer configuration with jumper A60 (3-4) installed on HPR05 PCB)
8	(Unused)		

Dipswitch Module B

#	Function		Meaning
1	Full Duplex	ON	Operates in full-duplex mode.
		OFF	Operates in half-duplex mode.
2	Parity Enable	ON	Enables parity checking and parity transmission.
3,5	BAUD		**3** / **5** / **Baud Rate** on / on / 110 on / off / 300 off / on / 1200 off / off / Option Baud Rate per switches 3, 4, & 5 on HPR05 PCB.
4	(Unused)		
6	Parity		This switch is used in conjunction with the parity enable switch.
		ON	Selects Even parity check and transmission
		OFF	Selects Odd parity check and transmission
7	Paper Out Defeat	ON	Paper Out sensing ignored.
8	(Unused)		

A66 Control Switch Functions

#	Function	Meaning
1	ETX/ACK	When this switch in ON, an ACK character will be transmitted whenever an ETX character is encountered in the print buffer. ETX characters are not printed. When the switch is OFF, ETX characters are ignored.
2	DC1/DC3	When this switch is ON, a DC3 code will be transmitted through the interface if printing is attempted.
3, 4, 5	BAUD	These three switches set the optional baud rate. When the two BAUD switches on the operator control panel are set to OFF, the baud rate selected by switches 3, 4, & 5 is used as the data communication speed. To prevent print buffer overflow, when operating at rates above 300 baud, the system must use DC1/DC3 or ETX/ACK protocol, or must monitor and respond to the Printer Ready interface line.

3	4	5	Baud Rate
off	off	off	150
on	off	off	600
off	on	off	1800
on	on	off	2000
off	off	on	2400
on	off	on	4800
off	on	on	7200
on	on	on	9600

A66 Control Switch Functions continuea

#	Function	Meaning
6, 7, 8	Font	These switches condition the printer to recognize a particular language font for data being received through the communications interface. Language font selection, whether by these three switches or by keyboard configuration, can be temporarily overridden by the sequence ESC SYN (n).

6	7	8	Meaning
off	off	off	Default Typewriter Paired
on	off	off	Typewriter Paired
off	on	off	Logical Bit Paired
on	on	off	APL
off	off	on	French AZERTY
on	off	on	German
off	on	on	Scandinavian
on	on	on	NORSK

Specifications

Print Speed:	Up to 40 characters per second with metalized printwheels.
Character Set:	88, 92, or 96 printable characters per printwheel. Switch-selectable program support for APL and all English language printwheels.
Printwheels:	88, 92, 96 character Xerox - Metal 96 character Diablo - Plastic
Character Spacing:	10-pitch = 10 characters/inch (3.94 ch/cm) 12-pitch = 12 characters/inch (4.72 ch/cm) 15-pitch = 15 characters/inch (5.91 ch/cm) Proportional Space (PS) - see HMI, page 241.
Column Spacing:	1/120 inch (.21mm) minimum.
Print Line:	13.2 inches (335.3mm) 132 columns 10-pitch 158 columns 12-pitch 198 columns 15-pitch
Print Buffer:	2688 bytes.
Paper Width:	16.53 inches (419.9mm) maximum - friction feed without Top Paper Out switch. 16.00 inches (406.4mm) maximum - friction feed with Top Paper Out switch. 15.25 inches (387.4mm) maximum - full width with optional forms tractor (14.75 inches/-374.7mm between holes). 3.25 inches (82.55mm) minimum with forms tractor (2.75 inches/69.85mm between holes).
Carriage Speed:	400 msec maximum for 13.1 inches (332.77mm) of motion.
Tabulation:	Left or right.
Line Spacing:	1/48 inch (.53mm) minimum.
Paper Feed:	Bidirectional, except with unidirectional forms tractor and unidirectional pin feed platen.

Paper Feed Speed:	4 inches (101.6mm) per second plus 40 msec (typical) settling delay time.
Paper Thickness:	.000 - .010 inch (.254mm) at low setting (1-3 part forms) .010 - .027 inch (.254 - .686mm) at high setting (4-6 part forms).
Sensors:	End of ribbon, paper out, and cover open.
Other Features:	Self test; host program control through escape sequences; data receive/transmit speed selection.
Power Requirements:	Strappable for operation from nominal 100, 120, 220, or 240 volt (+ 10%/-15%) AC inputs, 49-61 Hz. 350W maximum power consumption. Factory preset for 120 VAC. Check your printer's serial plate for proper input power.

Cabling Requirements

A standard RS-232-C interface cable is required for connection between the screen and the printer. This cable must be equipped with DB-25P connectors with the following pins connected:

PIN NO.	CCITT DESIG.	TELCO DESIG.	DESCRIPTION
1	101	AA	Protective Ground
2	103	BA	Transmitted Data
3	104	BB	Received Data
4	105	CA	Request To Send
6	107	CC	Data Set Ready *
7	102	AB	Signal Ground
20	108	CD	Data terminal Ready

* Pin 6 must be HI to receive or transmit data.

1.0 INTRODUCTION

The SA1403D Controller consists of a micrprocessor based controller with on-board data separator logic and is able to control a maximum of four drives. The drives can be any combination of Shugart SA1000 fixed disk drives, SA800 floppy disk drives, or SA850 floppy disk drives. The floppy disk track formats are compatible with IBM 1D/2D track formats. The SA1403D can be mounted on the SA1000 drive.

Commands are issued to the controller over a bidirectional bus connected to the host computer. The data separator/"serdes" logic serializes bytes and converts to FM/MFM data, and deserializes FM/MFM data into 8-bit bytes.

Due to the microprogrammed approach utilized in the controller, limited diagnostic capabilities are implemented. This methodology increases fault isolation efficiency and reduces system down time. Error detection and correction will tolerate media imperfections up to 4-bit burst errors.

NOTE: This device utilizes negative logic (i.e., 0V = logical 1)

1.1 SA1403D CONTROLLER FEATURES

OVERLAPPED SEEK	In multiple drive configurations the host can issue seeks to different drives without waiting for the first drive to complete its seek.
AUTOMATIC SEEK AND VERIFY	A seek command is implied in every data transfer command (READ, WRITE CHECK, etc.). If the heads are not positioned over the correct cylinder, a seek is initiated and a cylinder verification is performed after the seek completes.
FAULT DETECTION	Three classes of fault detection are provided for fault diagnosis: 1) Disk related faults. 2) Controller related faults. 3) Host command or I/O timing faults. Fault detection is available from the interface as a status message and is also visibly displayed on a row of status LED's on the controller PCB.
AUTOMATIC HEAD AND CYLINDER SWITCHING	If during a multi-block data transfer the end of a track is reached, the controller automatically switches to the next track. If the end of a cylinder is reached, the controller issues a seek and resumes the transfer.
DATA ERROR SENSING AND CORRECTION	If a data error is detected during a disk data transfer, the controller indicates whether or not it is correctable. If correctable, it can be automatically corrected. (This applies to the SA1000 only. CRC error detection is used on floppy disc drives.)
LOGICAL TO PHYSICAL DRIVE CORRELATION	Logical Unit Number (LUN's) are independent of physical port numbers. All accesses specify LUN's.
ON BOARD SECTOR BUFFER	A sector buffer is provided on the controller to eliminate the possibility of data overruns during a data transfer.
EFFICIENT HOST INTERFACE PROTOCOL	A bidirectional bus between the controller and host provides a simple, yet efficient communication path. In addition, a high level command set permits effective command initiation.
SECTOR INTERLEAVE	Sector interleaving is programmable with up to a 16 way interleave.
ODD PARITY	The 8 data bits on the interface bus can have odd parity. Depending on user preference, parity can be disabled.
FIXED SECTOR SIZE	The sector size is fixed at 256 bytes of data for the SA1000.

NUMBER OF DRIVES — The controller will connect to a maximum of four (4) drives. The drives can be any combination of SA1000's and/or SA850's and/or SA800's

1.1.1 OPTIONAL FEATURES

MICRO DIAGNOSTICS — A set of diagnostic PROM's are available to allow stand alone diagnostic testing of both drive and controller. Reference Appendix A.

1.1.2 SYSTEM CONFIGURATION

The controller and data separator comprise a single PCB that can be mounted onto the SA1000 drive. A maximum of four (4) drives may be connected as shown in Figure 2.

1.2 TRACK FORMATS AND CAPACITY

A) 32 sectors of 256 bytes per sector (SA1000only).
C) 26 sectors of 256 bytes per sector (Floppy only).
D) 26 sectors of 128 bytes per sector (Floppy only.)

IBM 1D/2D TRACK FORMAT — Track format for Floppy Disk drives can be selected under program control in real time. The track formats are:

1) Single density, single sided
2) Single density, double sided
3) Double density, single sided
4) Double density, double sided

	26 SECTOR	32 SECTOR
SA800	2001	N/A
SA850	4003	N/A
SA1002	N/A	16383
SA1004	N/A	32767

TABLE I.

Format/Capacity Relationship
Maximum Logical Sector Address Shown

2.0 SPECIFICATION SUMMARY

2.1 ENVIRONMENTAL LIMITS

	Operating	Storage
Temperature F/C	32°/0° to 131°/55°	-40°/-40° to 167°/75°
Max. Wet Bulb	85°F	non condensing
Relative Humidity	10% to 95%	10% to 95%
Altitude	Sea level to 10,000 ft	Sea level to 15,000 ft

2.2 POWER REQUIREMENTS

Three power supply voltages are required for the SA1400 series controllers. The maximum current requirements are as follows:

+5VDC ± 5% at 4.6 Amps
-5VDC ± 5% at 0.5 Amps
+24VDC ± 10% at 0.1 Amps

Power is applied to the SA1400 series controller via J10 which is a 6 pin AMP Mate-N-Lok connector (P/N 1-380999-0) mounted on the component side of the board. The recommended mating connector, P10, is an AMP P/N 1-480270-0 utilizing AMP pins P/N 60619-1. The J10 pins are labeled on the connector. Figure 1 shows the pin assignments.

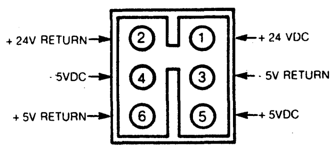


FIGURE 1. J10 DC POWER CONNECTOR

2.3 PHYSICAL PARAMETERS

Length:	13.7 inches (34.8cm) ± .030" (.076 cm)
Width:	8.25 inches (21cm) ± .010" (.025 cm)
Height:	0.5 inches (1.3cm) ± .030" (.076 cm)
Weight:	1.12 lbs (0.5Kg) ± .010 lbs (0.25 g)

3.0 SA1403D DISK DRIVE INTERFACE

Shugart SA1000 and SA800/850 disk drives are interfaced to the controller via J1, J2, J3, J4 and J5. Refer to Figure 2 for connection block diagram.

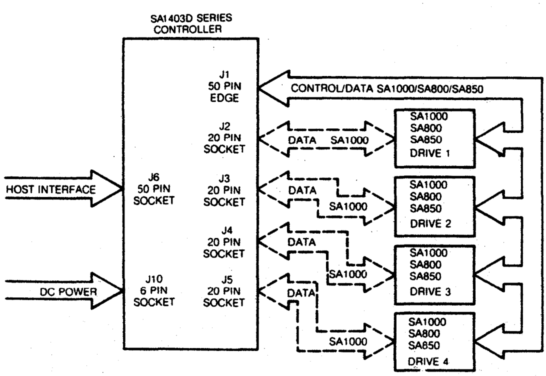


FIGURE 2. SA1403D INTERCONNECT DIAGRAM

NOTE: The last physical device on the control cable (drive to be terminated) must be an SA1000.

J1 is a 50 pin edge type connector which connects all drives in a daisy chain configuration. This connector carries control and data information for the floppy disk drives and control information only for the SA1000 disk drive. Maximum cable length should not exceed 20 feet (6 meters).

The recommended mating connector for J1 is a 3M Scotchflex ribbon connector P/N 3415-0001.

J2 through J5 are 20 pin socket type connectors used to radially connect the SA1000 data lines to the controller. Maximum cable length should not exceed 20 feet (6 meters).

The recommended mating connector for J2 through J5 is a 3M Scotchflex P/N 3421-3000. Figure 3 shows the pinouts for J1 and J2 through J5.

J1		SA1000	SA800	SA850
1	2	- IW SWITCH		- IW SWITCH
	4			
	6			
	8	- SEEK COMPLETE		
	10			- TWO SIDED
	12			
	14	- HEAD SEL 2^0		- SIDE SEL
	16			
	18	- HEAD SEL 2^1	- HEAD LOAD	- HEAD LOAD
	20	- INDEX	- INDEX	- INDEX
	22	- READY	- READY	- READY
	24			
	26	- DRIVE SEL 1	- DRIVE SEL 1	- DRIVE SEL 1
	28	- DRIVE SEL 2	- DRIVE SEL 2	- DRIVE SEL 2
	30	- DRIVE SEL 3	- DRIVE SEL 3	- DRIVE SEL 3
	32	- DRIVE SEL 4	- DRIVE SEL 4	- DRIVE SEL 4
	34	- DIRECTION SEL	- DIRECTION SEL	- DIRECTION SEL
	36	- STEP	- STEP	- STEP
	38		- WRITE DATA	- WRITE DATA
	40	- WRITE GATE	- WRITE GATE	- WRITE GATE
	42	- TRACK 000	- TRACK 00	- TRACK 00
	44	- WRITE FAULT	- WRITE PROTECT	- WRITE PROTECT
	46		- READ DATA	- READ DATA
	48			
49	50			

J2 THROUGH J5

- DRIVE SELECTED	1	2	GROUND
	3	4	
	5	6	
	7	8	GROUND
+ TIMING CLOCK	9	10	- TIMING CLOCK
GROUND	11	12	GROUND
+ MFM WRITE DATA	13	14	- MFM WRITE DATA
GROUND	15	16	GROUND
+ MFM READ DATA	17	18	- MFM READ DATA
GROUND	19	20	GROUND

FIGURE 3. SA1403D DRIVE CONNECTOR PINOUTS

3.1 CABLE TERMINATION

The last physical drive at the end of J1 (50 pin) cable must be properly terminated. Termination networks are provided on the drives (refer to SA1000, SA800 or SA850 OEM manuals for location of termination networks). Termination networks must be removed from all drives except the last drive on the cable to avoid multiple termination.

NOTE: If a combination of fixed and floppy drive are used,.the last drive at the end of the control cable must be an SA1000.

4.0 HOST CPU INTERFACE

The SA1400 series controller interface is a general purpose 8 bit parallel DMA.

The Host CPU is interfaced to the controller via connector J6, J6 is a 50 pin socket type connector. The recommended mating connector for J6 is a 3M Scotchflex ribbon connector P/N 3425-3000. The J6 interface cable should not exceed 20 feet (6 meters).

4.1 HOST CPU ELECTRICAL INTERFACE

All Host CPU interface signals are negative true. The signals are "Asserted" at 0 VDC to 0.4 VDC. The signals are "Deasserted" or inactive at 2.5 VDC to 5.25 VDC.

4.1.1 HOST CPU INTERFACE TERMINATION

All Host CPU interface timing lines are terminated with a 220/330 ohm network. The Host CPU adapter should be terminated in a similar fashion (see Figure 4).

The devices driving the controller inputs should be open collector devices capable of sinking at least 48 milliamps to a voltage level of less than 0.5 VDC (7438 or equivalent).

The devices receiving the controller outputs should be of the SCHMITT trigger type to improve the noise margin (74LS240, 74LS14, or equivalent). The Host adaptor should not load the bus with more than 1 standard TTL input load per line.

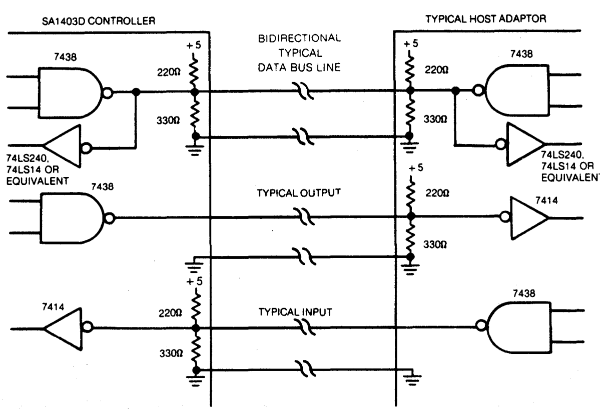
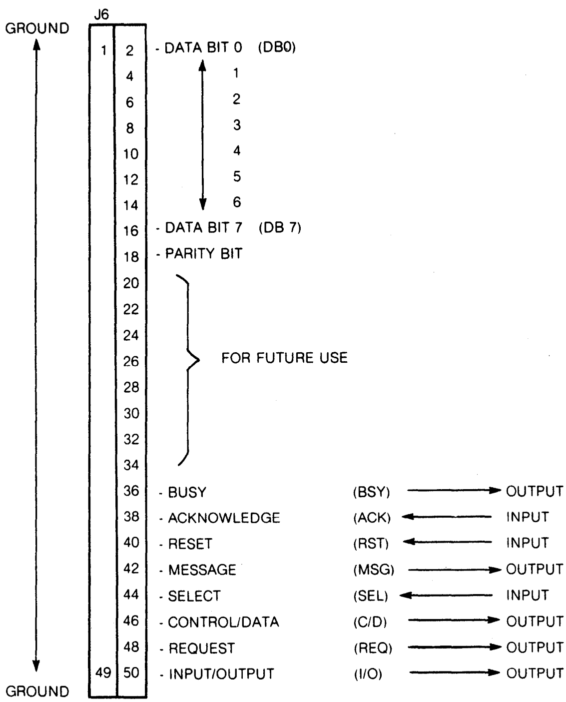


FIGURE 4. HOST ADAPTOR BUS TERMINATION

4.1.2 HOST CPU SIGNAL INTERFACE

The Host CPU signals are interfaced via J6. See figure 5 for J6 pinouts.



NOTE: ALL SIGNALS ARE TTL NEGATIVE TRUE

FIGURE 5. J6 HOST INTERFACE CONNECTOR PINOUT

4.2 SA1403D HOST BUS

4.2.1 THEORY OF OPERATIONS

Disk commands are issued to the SA1403D via the host bus following a defined protocol. The host initiates a command sequence by selecting the controller on the bus. If the controller is not busy, it requests command bytes from the host for task execution. (Command structure is described in 4.5). Depending on the type of command, the controller will request either 6 or 10 bytes. Upon reception of the last command byte, the controller begins execution of the command.

For the data transfer commands. a check is performed on the disk address and status flagged if it exceeds the drive limits. The data is stored in a sector buffer before transfer to the host or disk drive. This buffer eliminates any possibility of data overruns between the host and the disk.

Upon completion of the command, the controller will send completion status to the host. Further delineation of the completion status may be requested by issuing the appropriate sense commands.

Odd parity is generated by the SA1403D for all information that it puts on the I/O bus. If enabled, the SA1403D checks all information that it receives for odd parity.

4.3 SIGNAL DEFINITION

Unidirectional Signals Driven By Controller

I/O **Input/Output.** When asserted, the data on the bus is driven by the controller; when deasserted, the data on the bus is driven by the host adapter. The host adapter will use this line to enable its drivers onto the data bus.

C/D **Control/Data.** When asserted the data transmitted across the bus will be the command or status bytes; when deasserted the data will be the disk data bytes.

BUSY This bit is asserted as a response to the SEL line from the host adapter and to indicate that the host bus is currently in use.

MSG **Message.** When asserted indicates that the command is completed and status has been transferred. The assertion of this bit is always followed with the assertion of I/O, and the assertion of REQ, to cause a message byte transfer.

REQ **Request.** This bit operates in conjunction with I/O, C/D, & MSG. When asserted and I/O is asserted, REQ will mean that the data on the host bus is driven by the controller. When asserted and I/O is deasserted, REQ will mean that the data is driven by the host adaptor (H/A).

I/O	C/D	MSG	Meaning
d	a	d	Get command from H/A
d	d	d	Get data from H/A
a	d	d	Send data to H/A
a	a	d	Send status byte to H/A
a	a	a	Command done to H/A

TABLE 2.

a = asserted, d = deasserted, H/A = host adaptor

4.4 UNIDIRECTIONAL SIGNALS DRIVEN BY HOST ADAPTOR

ACK **Acknowledge.** This bit is asserted as a response to REQ from the controller. The timing requirements on this signal with respect to the data is described in REQuest section. ACK must be returned for each REQ assertion

RST **Reset.** Assertion by the Host causes the controller to cease all operations and return to an idle condition. This signal is normally used during a power up sequence. A reset during a write operation would cause incorrect data to be written on the selected disk. The controller may take a maximum of 2 seconds to respond to the select sequence following deassertion of the RESET line.

SEL **Select.** When asserted indicates the beginning of the command transaction. The H/A asserts SEL to gain the attention of the controller. Data bit zero on the host bus must also be asserted during SEL time to select the controller address. The controller will return BUSY within approximately 1μs.

4.4.1 DATA BUS BITS 0-7 (DB)

These bidirectional data lines are used to transfer 8 bit parallel data to/from the Host adaptor. Bit 7 is most signifant bit. NOTE: All I/F lines utilize negative logic.

4.4.2 PARITY BIT

This bit is asserted to maintain odd parity on all data and status information transfered to the Host. If enabled, the controller will test for odd parity on all command and data information transfered to the controller (see section 9.1).

4.5 HOST INTERFACE PROTOCOL

There are 4 sequences required to initiate and complete a command to the SA1403D series controller:

1) Controller Selection Sequence

2) Command Transfer Squence

3) Data Transfer Sequence

4) Status and Message Transfer Sequence

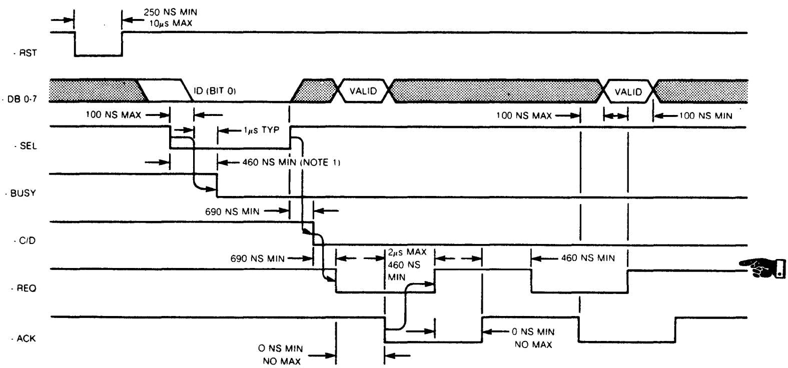
4.5.1 CONTROLLER SELECTION SEQUENCE

In order to gain the attention of the controller it is necessary to perform a selection sequence. Refer also to Figure 6.

The Host must first test BSY to determine if the controller is available. If BSY is deasserted, the Host will assert data bit 0 (controller ID) and then assert SEL. The controller will then respond by asserting BSY. At this point the Host must deassert SEL and data bit 0. I/O will remain deasserted throughout the selection sequence.

4.5.2 COMMAND TRANSFER SEQUENCE

Following the selection sequence the controller will assert REQ (see Figure 6). The Host will then place the first byte of the command descriptor block (see section 5.0) on the data bus. The Host will then assert ACK (if ACK is not asserted within 256 microseconds after the assertion of REQ, the controller will abort the command transfer sequence and attempt to transfer a status byte). The controller will respond by reading the byte on the data bus and then deasserting REQ. The Host then must deassert ACK to begin the next REQ/ACK handshake. This handshake must be completed to assure that all command and data bytes are transferred.



NOTE 1 - 2 SEC IMMEDIATELY AFTER RESET

FIGURE 6. SELECT SEQUENCE TIMING

4.5.3 DATA TRANSFER SEQUENCE

Following the command transfer sequence, the controller will respond on one of four ways:

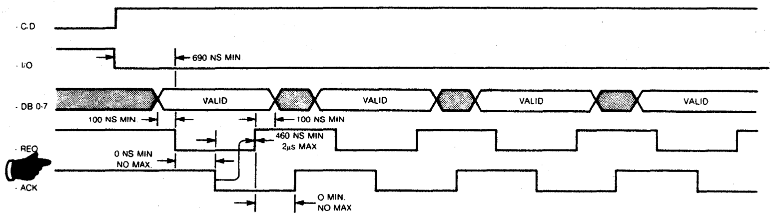
1) Begin seeking the drive.

2) Begin accepting write data from the Host.

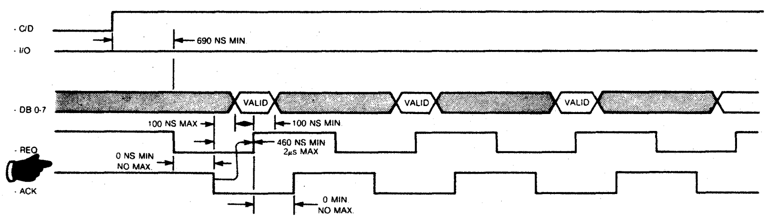
3) Begin transferring read data to the Host.

4) Return status to the Host.

If the command sent to the controller involves a data transfer (see Figure 7), the controller will deassert the C/D line to indicate a data transfer. If the data transfer is from the Host to the controller (write data) the I/O line will be deasserted. If the data transfer is from the controller to the Host (read data) the I/O line will be asserted. The controller will then set the REQ line to request a byte transfer. The Host will respond by transferring a byte across the data bus and then asserting ACK (if ACK is not asserted within 256 microseconds after the assertion of REQ, the controller will abort the data transfer sequence and attempt to transfer a status byte - see section 4.5.4). The Host will then deassert ACK and wait for the next assertion of REQ. This handshake continues until all data has been transferred



READ DATA TRANSFER SEQUENCE (CONTROLLER TO HOST)



WRITE DATA TRANSFER SEQUENCE (HOST TO CONTROLLER)

FIGURE 7. DATA TRANSFER SEQUENCE TIMING

4.5.4 STATUS AND MESSAGE TRANSFER SEQUENCE

Following a command transfer or data transfer, the controller will initiate a status byte and completion message transfer.

When a status byte transfer is required, the controller will assert C/D and I/O (see Figure 8). The controller will then assert REQ. The Host must then read the status byte on the data bus and then assert ACK (if ACK is not asserted within 256 microseconds after the assertion of REQ, REQ will be deasserted. REQ will then be asserted again).The controller will then deassert REQ. The host will then deassert ACK.

Following the status byte transfer, a completion message byte of all zero's will be transfered to indicate operation complete. The controller will assert the MSG line (along with I/O and C/D) and then assert REQ. The Host may read the completion message byte on the data bus and assert ACK (if ACK is not asserted within 256 microseconds, the controller will deassert the MSG line and attempt to transfer a status byte). The controller will respond by deasserting REQ. The Host will then deassert ACK. At this point BSY and all other controller I/O lines will be deasserted and the controller will return to an IDLE LOOP awaiting the next selection sequence.

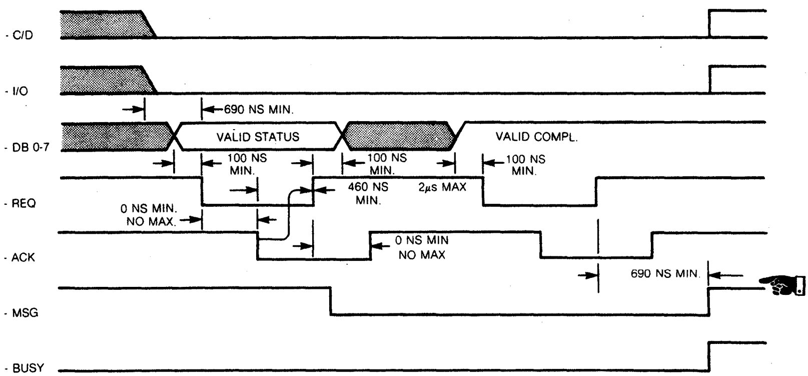


FIGURE 8. STATUS AND COMPLETION SEQUENCE TIMING

5.0 CONTROLLER COMMAND DESCRIPTOR BLOCK

Following the controller selection sequence the controller will request a command descriptor block (CDB) which, depending on the class of command, may be either 6 or 10 bytes in length. The first byte of the CDB contains the command class and the command operation code. The remaining bytes specify the drive logical unit number (LUN), logical sector address, number of sectors to be transfered or a destination device (Copy Command), and a control field byte.

Commands are categorized into four classes as indicated:

Class 0 - Utility, Data Transfer and Status Commands
Class 1 - Disk Copy Commands
Class 2-5,7 - Reserved
Class 6 - Floppy Disk Track Format Selection

The command descriptor blocks in Command Class 0 and 6 are 6 bytes long, and those in Class 1 are 10 bytes long.

The controller will check all incoming command descriptor blocks for validity and will also check (if enabled) all CDB's and data for odd parity (see section 9.1). A parity error will cause an immediate halt of the command or data transfer. This will not cause incorrect data to be written because the write does not occur until the sector buffer has been filled. An error in the command structure will cause a status byte transfer to occur upon completion of the CDB transfer.

5.1 COMMAND DESCRIPTION (CLASS 0)

WARNING!
Commands READ and WRITE require that the floppy diskette used be formatted. If unformatted, the controller will appear to "hang" - i.e., continue waiting for a data address mark. (Reset to clear this condition if it should occur).

Opcode (Hex)	**Description**
00	Test drive ready - Selects the drive and verifies drive ready. The ready condition is indicated by the status byte. A not-ready drive will cause bit 1 of the status byte to be set.
01	Recalibrate. Positions the R/W of selected drive arm to Track 00, clears error status in the drive.
02	Request Syndrome - returns two bytes of error offset and syndrom to the Host System for Host error correction capability (see Table 3). The first byte is offset in the data field of the error location. The most significant 3 bits of the second byte point to the beginning of the error location. The least significant 4 bits of the second byte are the syndrome which is a data correction mark to be exclusive or'ed with the faulty data. This command is only valid of the automatic data correction has been disabled.

	MSB 7	6	5	4	3	2	1	0 LSB
BYTE 1	BYTE OFFSET							
BYTE 2	BIT OFFSET			0	SYNDROME			

TABLE 3

03	Request Sense. This command must be issued immediately after an error. It returns 4 bytes of drive and controller sense for the specified LUN. (See copy block for exception)
04	Format Drive. Formats all blocks with ID field set according to interleave code. The data field contains E5 Hex.
05	Spare.
06	Format Track. *Formats the specified track with bad block flag cleared in all blocks of that track. Writes E5 Hex in the data fields.
07	Format Bad Track *(bad block flag). Formats the specified track with bad block flag set in the ID fields (bit 7 of the Head Address byte set). Writes E5 Hex in the data fields.
08	Read. Reads the specified number of blocks starting from initial block address given in the CDB. (See Warning above!)
09	Reserved.
0A	Write. Writes the specified number of blocks starting from initial block address given in the CDB. (See Warning above!)
0B	Seek. Initiates seek to specified block and immediately returns completion status before the seek is complete for those drives capable of overlap seek.

The track is addressed via the logical sector address, which may be any address within the desired track.

5.1.2 COMMAND DESCRIPTION (CLASS 1)

Opcode (Hex)	Description
00	Copy Blocks. Copies the specified number of blocks from Source LUN starting at the specified Logical address to Destination LUN starting at the specified Logical address. The number of sectors transferred may be from 1 to 256. The completion status byte will indicate the source LUN. If an error occurs, a Request Sense command is issued to the source LUN. The sense will indicate the type of error for the appropriate LUN. Note the data in the blocks will be truncated or appended with undefined data if the Source and Destination block sizes are not the same (e.g. Source block size - 128 bytes/sector, and Destination block size - 256 bytes/sector).

5.1.3 COMMAND DESCRIPTION (CLASS 6)

Opcode (Hex) **Description**

00 Define Floppy Disk Track Format. The Track format code in byte 6 of the CDB defines the track format for the LUN. The Track Format Codes are as follows:

Track Format Code (Hex)	Description
00	Single Density, Single Sided. All tracks - FM recording, 128 bytes/sector, 26 sectors/track.
01	Single Density, Double Sided. All tracks - FM recording, 128 bytes/sector, 26 sectors/track.
02	Double Density, Single Sided. Side 0, Cylinder 0 - FM Recording, 128 bytes/sector, 26 sectors/track. All other tracks - MFM recording, 256 bytes/sector, 26 sectors/track.
03	Double Density, Double Sided. Side 0, Cylinder 0 - FM recording, 128 bytes/sector, 26 sectors/track. All other track - MFM recording, 256 bytes/sector, 26 sectors/track.

NOTE: If track format information for floppy is not specified after each reset or power-on, the default mode will be taken from the drive type selection dipswitch as follows:

Switch Setting	Mode
OFF-ON	Single density, single sided (same as track format code 00)
OFF-OFF	Single density, double sided (same as track format code 01)

Refer to Section 9.2 for switch setup instructions.

5.2 COMMAND FORMAT

5.2.1 CLASS 0 COMMANDS

	7	6	5	4	3	2	1	0
byte #1	*0 0 0*			*opcode*				
byte #2	*LUN*			*logical adr2*** *(MS)*				
byte #3	*logical adr1***							
byte #4	*logical adr0*** *(LS)*							
byte #5	*number of blocks**							
byte #6	*control****							

* *Interleave factor for Format, Check Track Format commands.*
**Refer to Section 5.5 Logical Address.
****The control field is defined as follows:*

7	6	5	4	3	2	1	0

spare (set to zero) (bits 5–0)
disable data error correction = 1 (bit 6)
disable retry = 1 (bit 7)

CONTROL FIELD

5.2.2 CLASS 1 COMMANDS

	7	6	5	4	3	2	1	0
byte #1	0	0	1	opcode				
byte #2	0	LUN/s		logical adr2/s* (MS)				
byte #3	logical adr1/s*							
byte #4	logical adr0/s* (LS)							
byte #5	number of blocks							
byte #6	0	LUN/d		logical adr2/d* (MS)				
byte #7	logical adr1/d*							
byte #8	logical adr0/d* (LS)							
byte #9	spare							
byte #10	control (section 5.2.1)							

where 's' indicates the source device and 'd' indicates the destination device.
*Refer to Section 5.5 Logical Address

5.2.3 CLASS 6 COMMANDS

	7	6	5	4	3	2	1	0
byte #1	1	1	0	opcode				
byte #2	LUN			N/A				
byte #3	N/A							
byte #4	N/A							
byte #5	N/A							
byte #6	Track Format Code							

NOTE: See Class 6 Command Description for more information and default modes for floppy drives.

5.3 STATUS FORMAT

5.3.1 Completion Status Byte Format

At the normal termination of a command or following a fatal error, the controller will cause a status byte to be transferred from the controller to the Host. Bit 0, the least significant bit of the status byte, will be set equal to 1 if the controller detects a parity error during a command or data transfer to the controller. Bit 1 will be set = 1 if the controller detects an error condition. Bits 5 and 6 represent the LUN of the device where the error occured. If no error occurs, bit 0 - 4 will be set equal to 0.

Following the transfer of the status byte, the MSG line will be asserted to indicate a completion message. At this time the message consists of a single byte transfer with all bits set = 0.

Prior to an error condition the controller, unless diabled (see section 5.2.1 Control Field), will retry 3 times before posting the error.

7	6	5	4	3	2	1	0
0	MS LUN	LS LUN	spare (set = 0)			Error	Parity error

Bit 0 Parity error during transfer from host to controller.
Bit 1 Error occured during command execution.
Bit 2-4 Spare (set to zero).
Bit 5-7 Logical unit number of the drive.

5.3.2 DRIVE AND CONTROLLER SENSE BLOCK

Following an error indication from the status byte, the Host may perform a REQUEST SENSE command to obtain more detailed information about the error.

The REQUEST SENSE command will transfer a block of 4 bytes to the Host system.

	7	6	5	4	3	2	1	0
byte #1	Block address valid	spare (set to zero)	Error type		Error code			

	7	6	5	4	3	2	1	0
byte #2	LUN			logical adr2*				
byte #3	logical adr1*							
byte #4	logical adr0*							

*Refer to Section 5.5 Logical Address

5.4 ERROR CODES

5.4.1 TYPE 0 (DRIVE) ERROR CODES

0	No error
1	No Index signal
2	No Seek Complete
3	Write Fault (SA1000 only)
4	Drive not ready
5	Drive not selected (SA1000 only)
6	No Track 00

5.4.2 TYPE 1 (CONTROLLER) ERROR CODES

0	ID read error. ECC or CRC (floppy) error in the ID field (uncorrectable).
1	Uncorrectable data error during a read.
2	ID Address Mark not found (possibly unformated disk).
3	Data Address Mark not found.
4	Record not found. Found correct cylinder and head but not sector.
5	Seek error. R/W head positioned on a wrong cylinder and/or selected a wrong head.
6	DMA Data time out error. No Host acknowledge within 256μs.
7	Write protected. (SA800/850 only)
8	Correctable data field error. ECC error (automatic correction if not disabled).
9	Bad track found
A	Format Error. The controller detected that during the Check Track command, the format on the drive was not as expected.

5.4.3 TYPE 2 (COMMAND) ERROR CODES

0	Invalid Command received from the host.
1	Illegal logical sector address. Address is beyond the maximum address for the type of drive.
2	Illegal function for the specified drive.

5.5.4 TYPE 3 (MISC) ERROR CODES

0	RAM error. Data error detected during Sector buffer RAM diagnostic.

5.5 LOGICAL ADDRESS

The logical address is computed as follows:

Logical adr = (CYADR * HDCYL + HDADR) * SETRK + (SEADR)

Where:
CYADR = cylinder address
HDADR = head address
SEADR = sector address
HDCYL = number of heads per cylinder
SETRK = number of sectors per track

Bit 0 of Logical adr 0 = the least significant bit.
Bit 4 of Logical adr 2 = the most significant bit.

Note: All addresses begin with 00.

6.0 SECTOR INTERLEAVE CODES

In order to tailor host system data transfer speed to the disk rotational speed, sector interleaving is offered. Sixteen interleave codes are offered numbered 1 to 16. Not all interleave codes will result in optimum sector interleave, therefore the interleave should be chosen carefully. In order to maintain IBM floppy disk compatibility in interleave code of 1 should be used. This will result in a non-interleave condition.

6.1 SELECTING THE RIGID DISK INTERLEAVE CODE

The interleave code given during the format command is used to calculate the logical sector number for the rigid disk as follows: Logical Sector = (Physical Sector × Interleave code) (mod 32). Note: when the logical sector number exceeds 31 the next logical sector is the lowest available physical sector. This does not always create a true modulo function.

Two examples of interleave codes are shown:

Interleave code of 2:

Physical:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Logical:	0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
Physical:	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Logical:	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31

Interleave code of 11:

Physical:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Logical:	0	11	22	1	12	23	2	13	24	3	14	25	4	15	26	5
Physcial:	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Logical:	16	27	6	17	28	7	18	29	8	19	30	9	20	31	10	21

Code	Number of Disk Revolutions Required to Read One Track	Time available to Transfer one Byte of Data (including controller time)	Minimum Number of Idle Sectors Between Reads
11	3	4.7μs	2
8	4	7.0μs	3
6	6	9.4μs	4
5	7	11.7μs	5
4	8	16.4μs	7
3	11	23.4μs	10
2	16	35.1μs	15
1	32	72.5μs	31

• (for SA1400 series controllers operating with SA1000 series drives - double density, 32 sectors, 256 bytes/sector.)
Note: Other codes will work, but require more revolutions of the disk to read all sectors of one track.

TABLE 3. INTERLEAVE CODE SELECTION CHART*

7.0 DIAGNOSTIC PHILSOPHY

7.1 BOARD RESIDENT MICRODIAGNOSTIC

Fault Isolation Microdiagnostic (Optional)
The controller can be further checked out off-line by initiating explicit microdiagnostic routines via optional firmware diagnostic sets. The routines are initiated by a set of control switches. Errors will be dislayed in a set of LED's. Each microdiagnostic checks the funtionality of a particular section of the controller and is able to isolate failures in the following major categories:

- ALU
- Registers
- Sector Buffer
- ECC Logics

Fault-isolation techniques can be concentrated on the failing section.

8.0 STATUS LED ERROR INTERPRETATION

Drive/controller error conditions are displayed on the 8 LED display lights provided near the J10 DC power connector (see Figures 11). The following list of hexadecimal numbered error codes describe error meanings. Note that these error codes do not necessarily match the request sense block error codes. LED number 7 is the MSB.

01	No Index Detected
02	No Track Zero Detected
03	Illegal Logical Sector Address - beyond maximum sectors available for type of drive
04	Drive Not Selected (SA1000 only)
05	No Seek Complete Detected
06	ID Address Mark Not found (unformatted)
07	Data Address Mark Not found
08	Seek Error - R/W head not positioned on correct track
09	Record Not found - found correct cylinder and head but not sector
0A	ID ECC or CRC error (uncorrectable)
0B	DMA Timeout Error - no Host acknowledge within 256μsec after request.
0C	Invalid Command Received from Host
0D	Incorrect Data Address Mark
0E	Incorrect ID Address Mark
0F	Incorrect Cylinder Address
10	Incorrect Sector Address
11	Incorrect Head Address
12	Uncorrectable Data Field ECC or CRC error
13	Correctable Data Field ECC error
14	Drive Not Ready
15	Write Fault (SA1000 and SA4000/4100 only)
16	Spare
17	Write Protected (SA800/850 only)
18	RAM Diagnostic Error
19-1F	Spare
20	Parity Error
21	Bad Sector found - a sector within a track that has been flagged bad has been found.
22	Invalid function for this drive type.

9.0 CONTROLLER OPTION SELECTION

9.1 PARITY SELECT JUMPERS

Odd parity may be used by the Host system for data integrity verification. The controller will always output odd parity to the Host system.

Odd parity checking by the controller may be allowed or inhibited by moving a 3 position jumper plug at W2 located near the J6 Host connector (see Figure 11). With jumper at position A + B the controller will test for odd parity on all data input to the controller. With jumper at positon B + C the controller will not check for parity (normally shipped in A + B).

9.2 DRIVE TYPE SELECTION DIPSWITCH

The dipswitch settings for various types of drives for the SA1403D are shown below:

Prom Set AS30 — I, II, III, IV   CUSTOMER FIRMWARE: (DIP SWITCH set-up procedure)

Location: 2H

Switch Bits	8 7	6 5	4 3	2 1	
Field Definition	LUN 0 Drive Type	LUN 1 Drive Type	LUN 2 Drive Type	LUN 3 Drive Type	OFF ON

Drive Type	Switch Setting Even	Switch Setting Odd	Description	
0	on	on	SA1002	2 heads, 256 cylinders
1	on	off	SA1004	4 heads, 256 cylinders
2	off	on	SA800	1 head, 77 cylinders
3	off	off	SA850	2 heads, 77 cylinders

EXAMPLE:

LOCATION: 23

	8 7	6 5	4 3	2 1	
	LUN 0 Drive Type	LUN 1 Drive Type	LUN 2 Drive Type	LUN 3 Drive Type	OFF
	on on	off on	on off	off off	ON

Drive 0 is set up for SA1002
Drive 1 is set up for SA800
Drive 2 is set up for SA1004
Drive 3 is set up for SA850

10.0 TRACK FORMAT DESCRIPTION

10.1 26 SECTOR FORMAT

The 26 sector format is an IBM compatible format which employes FM single density encoding on all tracks of the single density format (IBM 3740 compatible) and on track 0, side 0 of the double density format. This format yields 26 sectors of 128 bytes per sector.

The remainder of the tracks on the double density formats are encoded with MFM double density which yields 26 sectors of 256 bytes per sector (IBM system 34 compatible). Figure 9 shows the two type of encoding utilized.

REPEATED 26 TIMES (188 BYTES)

	PRE INDEX GAP	INDEX SYNC	INDEX AM	GAP 1	ID SYNC	ID AM	TRK	HD	SEC	SEC/ LEN.	CRC	GAP 2	DATA SYNC	DATA AM	DATA FIELD	CRC	GAP 3	GAP 4
HEX DATA	FF	00	FC	FF	00	FE	X	0	X	0	X	FF	00	FB	X	X	FF	FF
NUM. OF BYTES	40	6	1	26	6	1	1	1	1	1	2	11	6	1	128	2	27	247 (NOM.)

WRITE UPDATE

FM FORMAT IBM 3740

REPEATED 26 TIMES (372 BYTES)

	PRE INDEX GAP	INDEX SYNC	PRE INDEX AM	INDEX AM	GAP 1	ID SYNC	PRE ID AM	ID AM	TRK	HD	SEC	SEC LEN	CRC	GAP 2	DATA SYNC	PRE DATA AM	DATA AM	DATA FIELD	CRC	GAP 3	GAP 4
HEX DATA	4E	00	A1	FC	4E	00	A1	FE	X	X	X	01	X	4E	00	A1	FB	X	X	4E	4E
NUM. OF BYTES	80	12	3	1	50	12	3	1	1	1	1	1	2	22		3		256	2	54	598 (NOM.)

WRITE UPDATE

MFM FORMAT IBM SYSTEM 34

FIGURE 9. 26 SECTOR FORMAT - SA800/850

10.2 32 SECTOR FORMAT

The 32 sector format employs MFM encoding on all tracks of the SA1000. This format yields 32 sectors of 256 bytes per sector. Figure 10 shows the 32 sector format.

REPEATED 32 TIMES (316 BYTES)

	GAP 1	SYNC	PRE ID AM	ID AM	CYL	H.I	SEC	ECC	GAP 2		PRE DATA AM	DATA AM	DATA FIELD	ECC	GAP 3		GAP 4
HEX DATA	4E	00	A1	FE	X	X	X	X	00	00	A1	F8	X	X	00	00	4E
NUM. OF BYTE!	16	13	1	1	1	1	1	3	2	13	1	1	256	3	2	15	352 (NOM.)

WRITE UPDATE

FIGURE 10. 32 SECTOR FORMAT - SA1000

11.0 DRIVE JUMPER SETTINGS

11.1 JUMPER SETTINGS FOR SA800/801 FLOPPY

The following information is contained in the SA800/801 Diskette Storage Drive OEM Manual, Shugart Associates, 1977.

Jumper Name	Function (Enabled if Jumper Installed)
A	Install enable DRSEL to drive selection
B	Install, Head Load on Drive Select
C	Remove, Drive Select loads heads
D	Remove, In Use to LED is disabled
DC	Remove, Disable Disk Change to return to controller
DS	Install enable stepper on Drive Select
DS1-4	Install one only, DS1 = LUN 0 (Drive Select)
HL	Remove, Head load on Drive Select
L	Jumper for -5V (remove for -15V), controller requires -5V only
T1	Remove, Head Load terminator
T2	Install, Pullup for Drive Select lines
T3	Install, Direction terminator
T4	Install, Step terminator
T5	Install, Write Data terminator
T6	Install, Write Gate terminator
X	Install, Head Load Enable
Y	Remove, Disable Hdld from driving LED
Z	Install drive select drives in use LED
800	Install, enables 800 index only operation
801	Remove, disables 801 mode operation

11.2 JUMPER SETTINGS FOR SA850/851 FLOPPY

Jumper Name **Function (Enabled if Jumper Installed)**

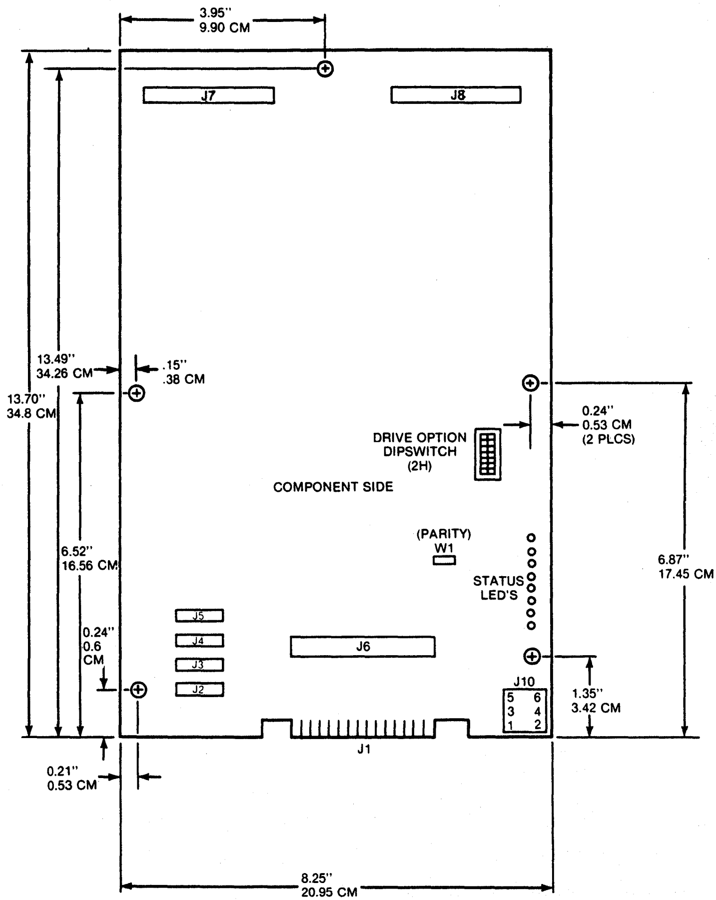
Controller is compatible with the factory jumper configuration. See SA850/851 OEM Manual.

Note: Jumpers must be set for SA850, not SA851

11.3 JUMPER SETTINGS FOR SA1000 WINCHESTER

Jumper Name **Function (Enabled if Jumper Installed)**

Controller is compatible with the factory jumper configuration. See SA1000 OEM Manual.



ALL TOLERANCES ± .003"

FIGURE 11. SA1403D DIMENSIONAL DRAWING

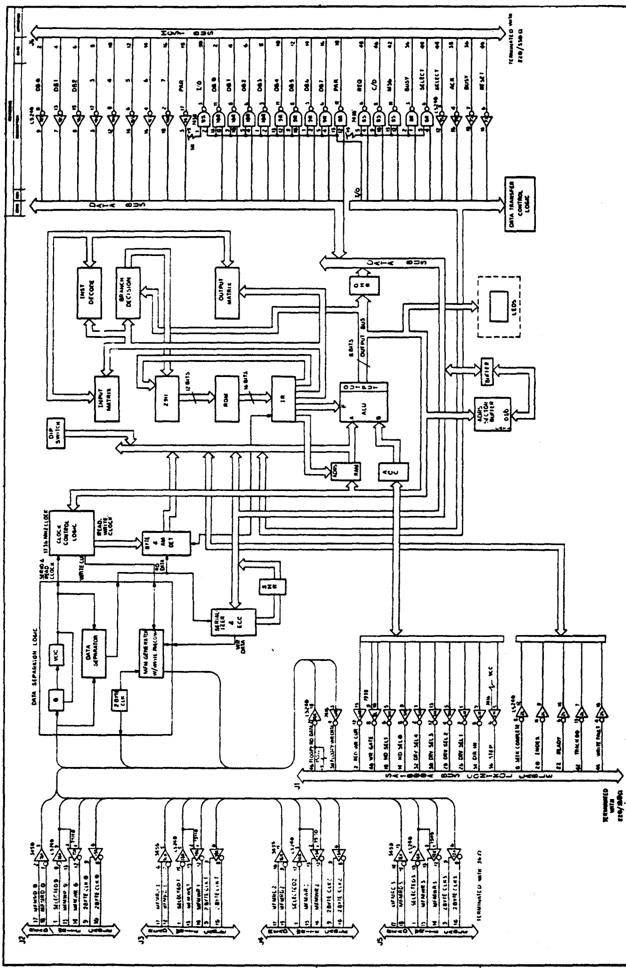


FIGURE 12. SA1403D FUNCTIONAL BLOCK DIAGRAM

Notes

WESTERN DIGITAL
C O R P O R A T I O N

FD179X-02
Floppy Disk Formatter/Controller Family

FEATURES

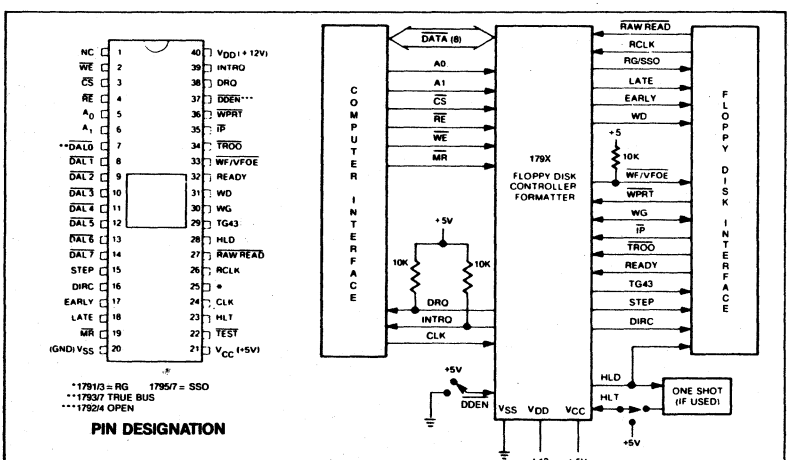
- TWO VFO CONTROL SIGNALS — RG & VFOE
- SOFT SECTOR FORMAT COMPATIBILITY
- AUTOMATIC TRACK SEEK WITH VERIFICATION
- ACCOMMODATES SINGLE AND DOUBLE DENSITY FORMATS
  - IBM 3740 Single Density (FM)
  - IBM System 34 Double Density (MFM)
  - Non IBM Format for Increased Capacity
- READ MODE
  - Single/Multiple Sector Read with Automatic Search or Entire Track Read
  - Selectable 128, 256, 512 or 1024 Byte Sector Lengths
- WRITE MODE
  - Single/Multiple Sector Write with Automatic Sector Search
  - Entire Track Write for Diskette Formatting
- SYSTEM COMPATIBILITY
  - Double Buffering of Data 8 Bit Bi-Directional Bus for Data, Control and Status
  - DMA or Programmed Data Transfers
  - All Inputs and Outputs are TTL Compatible
  - On-Chip Track and Sector Registers/Comprehensive Status Information
- PROGRAMMABLE CONTROLS
  - Selectable Track to Track Stepping Time
  - Side Select Compare
- INTERFACES TO WD1691 DATA SEPARATOR
- WINDOW EXTENSION
- INCORPORATES ENCODING/DECODING AND ADDRESS MARK CIRCUITRY
- FD1792/4 IS SINGLE DENSITY ONLY
- FD1795/7 HAS A SIDE SELECT OUTPUT

179X-02 FAMILY CHARACTERISTICS

FEATURES	1791	1792	1793	1794	1795	1797
Single Density (FM)	X	X	X	X	X	X
Double Density (MFM)	X		X		X	X
True Data Bus			X	X		X
Inverted Data Bus	X	X			X	
Write Precomp	X	X	X	X	X	X
Side Selection Output					X	X

APPLICATIONS

8" FLOPPY AND 5¼" MINI FLOPPY CONTROLLER
SINGLE OR DOUBLE DENSITY
CONTROLLER/FORMATTER



FD179X SYSTEM BLOCK DIAGRAM

PIN OUTS

PIN NUMBER	PIN NAME	SYMBOL	FUNCTION
1	NO CONNECTION	NC	Pin 1 is internally connected to a back bias generator and must be left open by the user.
19	$\overline{\text{MASTER RESET}}$	$\overline{\text{MR}}$	A logic low (50 microseconds min.) on this input resets the device and loads HEX 03 into the command register. The Not Ready (Status Bit 7) is reset during $\overline{\text{MR}}$ ACTIVE. When $\overline{\text{MR}}$ is brought to a logic high a RESTORE Command is executed, regardless of the state of the Ready signal from the drive. Also, HEX 01 is loaded into sector register.
20	POWER SUPPLIES	V_{SS}	Ground
21		V_{CC}	+5V ±5%
40		V_{DD}	+12V ±5%
COMPUTER INTERFACE:			
2	$\overline{\text{WRITE ENABLE}}$	$\overline{\text{WE}}$	A logic low on this input gates data on the DAL into the selected register when $\overline{\text{CS}}$ is low.
3	$\overline{\text{CHIP SELECT}}$	$\overline{\text{CS}}$	A logic low on this input selects the chip and enables computer communication with the device.
4	$\overline{\text{READ ENABLE}}$	$\overline{\text{RE}}$	A logic low on this input controls the placement of data from a selected register on the DAL when $\overline{\text{CS}}$ is low.
5,6	REGISTER SELECT LINES	A0, A1	These inputs select the register to receive/transfer data on the DAL lines under $\overline{\text{RE}}$ and $\overline{\text{WE}}$ control: $\overline{\text{CS}}$ A1 A0 — $\overline{\text{RE}}$ — $\overline{\text{WE}}$ 0 0 0 Status Reg Command Reg 0 0 1 Track Reg Track Reg 0 1 0 Sector Reg Sector Reg 0 1 1 Data Reg Data Reg
7-14	$\overline{\text{DATA ACCESS LINES}}$	$\overline{\text{DAL0-DAL7}}$	Eight bit Bidirectional bus used for transfer of data, control, and status. This bus is receiver enabled by $\overline{\text{WE}}$ or transmitter enabled by $\overline{\text{RE}}$. Each line will drive 1 standard TTL load.
24	CLOCK	CLK	This input requires a free-running 50% duty cycle square wave clock for internal timing reference, 2 MHz ± 1% for 8" drives, 1 MHz ± 1% for mini-floppies.
38	DATA REQUEST	DRQ	This open drain output indicates that the DR contains assembled data in Read operations, or the DR is empty in Write operations. This signal is reset when serviced by the computer through reading or loading the DR in Read or Write operations, respectively. Use 10K pull-up resistor to +5.
39	INTERRUPT REQUEST	INTRQ	This open drain output is set at the completion of any command and is reset when the STATUS register is read or the command register is written to. Use 10K pull-up resistor to +5.
FLOPPY DISK INTERFACE:			
15	STEP	STEP	The step output contains a pulse for each step.
16	DIRECTION	DIRC	Direction Output is active high when stepping in, active low when stepping out.
17	EARLY	EARLY	Indicates that the WRITE DATA pulse occuring while Early is active (high) should be shifted early for write precompensation.
18	LATE	LATE	Indicates that the write data pulse occurring while Late is active (high) should be shifted late for write precompensation.

PIN NUMBER	PIN NAME	SYMBOL	FUNCTION
22	$\overline{\text{TEST}}$	$\overline{\text{TEST}}$	This input is used for testing purposes only and should be tied to +5V or left open by the user unless interfacing to voice coil actuated steppers.
23	HEAD LOAD TIMING	HLT	When a logic high is found on the HLT input the head is assumed to be engaged. It is typically derived from a 1 shot triggered by HLD.
25	READ GATE (1791, 1792, 1793, 1794)	RG	This output is used for synchronization of external data separators. The output goes high after two Bytes of zeros in single density, or 4 Bytes of either zeros or ones in double density operation.
25	SIDE SELECT OUTPUT (1795, 1797)	SSO	The logic level of the Side Select Output is directly controlled by the 'S' flag in Type II or III commands. When U = 1, SSO is set to a logic 1. When U = 0, SSO is set to a logic 0. The SSO is compared with the side information in the Sector I.D. Field. If they do not compare Status Bit 4 (RNF) is set. The Side Select Output is only updated at the beginning of a Type II or III command. It is forced to a logic 0 upon a MASTER RESET condition.
26	READ CLOCK	RCLK	A nominal square-wave clock signal derived from the data stream must be provided to this input. Phasing (i.e. RCLK transitions) relative to RAW READ is important but polarity (RCLK high or low) is not.
27	$\overline{\text{RAW READ}}$	$\overline{\text{RAW READ}}$	The data input signal directly from the drive. This input shall be a negative pulse for each recorded flux transition.
28	HEAD LOAD	HLD	The HLD output controls the loading of the Read-Write head against the media.
29	TRACK GREATER THAN 43	TG43	This output informs the drive that the Read/Write head is positioned between tracks 44-76. This output is valid only during Read and Write Commands.
30	WRITE GATE	WG	This output is made valid before writing is to be performed on the diskette.
31	WRITE DATA	WD	A 200 ns (MFM) or 500 ns (FM) output pulse per flux transition. WD contains the unique Address marks as well as data and clock in both FM and MFM formats.
32	READY	READY	This input indicates disk readiness and is sampled for a logic high before Read or Write commands are performed. If Ready is low the Read or Write operation is not performed and an interrupt is generated. Type I operations are performed regardless of the state of Ready. The Ready input appears in inverted format as Status Register bit 7.
33	$\overline{\text{WRITE FAULT}}$ $\overline{\text{VFO ENABLE}}$	$\overline{\text{WF/VFOE}}$	This is a bi-directional signal used to signify writing faults at the drive, and to enable the external PLO data separator. When WG = 1, Pin 33 functions as a WF input. If WF = 0, any write command will immediately be terminated. When WG = 0, Pin 33 functions as a VFOE output. VFOE will go low during a read operation after the head has loaded and settled (HLT = 1). On the 1795/7, it will remain low until the last bit of the second CRC byte in the ID field. VFOE will then go high until 8 bytes (MFM) or 4 bytes (FM) before the Address Mark. It will then go active until the last bit of the second CRC byte of the Data Field. On the 1791/3, VFOE will remain low until the end of the Data Field. This pin has an internal 100K Ohm pull-up resistor.
34	$\overline{\text{TRACK 00}}$	$\overline{\text{TR00}}$	This input informs the FD179X that the Read/Write head is positioned over Track 00.

PIN NUMBER	PIN NAME	SYMBOL	FUNCTION
35	$\overline{\text{INDEX PULSE}}$	$\overline{\text{IP}}$	This input informs the FD179X when the index hole is encountered on the diskette.
36	$\overline{\text{WRITE PROTECT}}$	$\overline{\text{WPRT}}$	This input is sampled whenever a Write Command is received. A logic low terminates the command and sets the Write Protect Status bit.
37	$\overline{\text{DOUBLE DENSITY}}$	$\overline{\text{DDEN}}$	This input pin selects either single or double density operation. When $\overline{\text{DDEN}}$ = 0, double density is selected. When $\overline{\text{DDEN}}$ = 1, single density is selected. This line must be left open on the 1792/4.

GENERAL DESCRIPTION

The FD179X are N-Channel Silicon Gate MOS LSI devices which perform the functions of a Floppy Disk Formatter/Controller in a single chip implementation. The FD179X, which can be considered the end result of both the FD1771 and FD1781 designs, is IBM 3740 compatible in single density mode (FM) and System 34 compatible in Double Density Mode (MFM). The FD179X contains all the features of its predecessor the FD1771, plus the added features necessary to read/write and format a double density diskette. These include address mark detection, FM and MFM encode and decode logic, window extension, and write precompensation. In order to maintain compatibility, the FD1771, FD1781, and FD179X designs were made as close as possible with the computer interface, instruction set, and I/O registers being identical. Also, head load control is identical. In each case, the actual pin assignments vary by only a few pins from any one to another.

The processor interface consists of an 8-bit bi-directional bus for data, status, and control word transfers. The FD179X is set up to operate on a multiplexed bus with other bus-oriented devices.

The FD179X is TTL compatible on all inputs and outputs. The outputs will drive ONE TTL load or three LS loads. The 1793 is identical to the 1791 except the DAL lines are TRUE for systems that utilize true data busses.

The 1795/7 has a side select output for controlling double sided drives, and the 1792 and 1794 are "Single Density Only" versions of the 1791 and 1793 respectively. On these devices, DDEN must be left open.

ORGANIZATION

The Floppy Disk Formatter block diagram is illustrated on page 5. The primary sections include the parallel processor interface and the Floppy Disk interface.

Data Shift Register — This 8-bit register assembles serial data from the Read Data input ($\overline{\text{RAW READ}}$) during Read operations and transfers serial data to the Write Data output during Write operations.

Data Register — This 8-bit register is used as a holding register during Disk Read and Write operations. In Disk Read operations the assembled data byte is transferred in parallel to the Data Register from the Data Shift Register. In Disk Write operations information is transferred in parallel from the Data Register to the Data Shift Register.

When executing the Seek command the Data Register holds the address of the desired Track position. This register is loaded from the DAL and gated onto the DAL under processor control.

Track Register — This 8-bit register holds the track number of the current Read/Write head position. It is incremented by one every time the head is stepped in (towards track 76) and decremented by one when the head is stepped out (towards track 00). The contents of the register are compared with the recorded track number in the ID field during disk Read, Write, and Verify operations. The Track Register can be loaded from or transferred to the DAL. This Register should not be loaded when the device is busy.

Sector Register (SR) — This 8-bit register holds the address of the desired sector position. The contents of the register are compared with the recorded sector number in the ID field during disk Read or Write operations. The Sector Register contents can be loaded from or transferred to the DAL. This register should not be loaded when the device is busy.

Command Register (CR) — This 8-bit register holds the command presently being executed. This register should not be loaded when the device is busy unless the new command is a force interrupt. The command register can be loaded from the DAL, but not read onto the DAL.

Status Register (STR) — This 8-bit register holds device Status information. The meaning of the Status bits is a function of the type of command previously executed. This register can be read onto the DAL, but not loaded from the DAL.

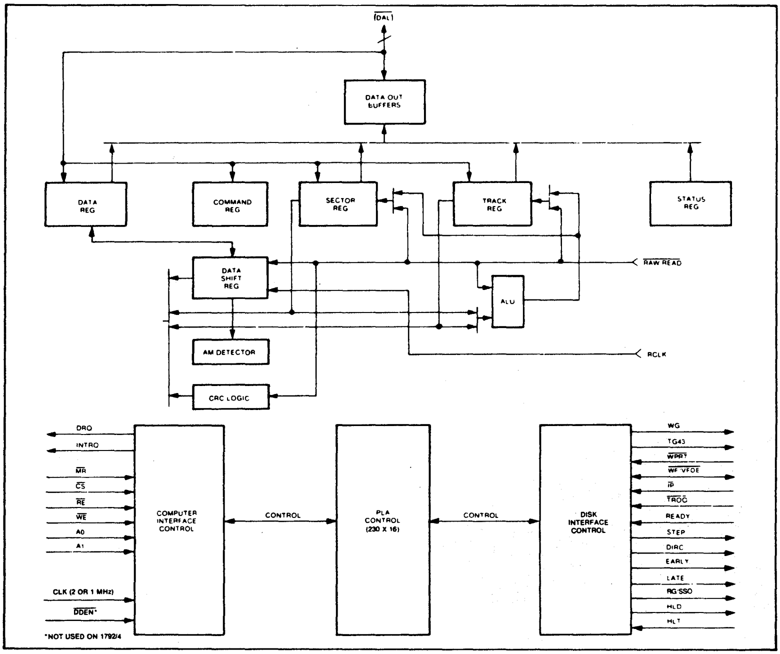
CRC Logic — This logic is used to check or to generate the 16-bit Cyclic Redundancy Check (CRC). The polynomial is: $G(x) = x^{16} + x^{12} + x^5 + 1$.

The CRC includes all information starting with the address mark and up to the CRC characters. The CRC register is preset to ones prior to data being shifted through the circuit.

Arithmetic/Logic Unit (ALU) — The ALU is a serial comparator, incrementer, and decrementer and is used for register modification and comparisons with the disk recorded ID field.

Timing and Control — All computer and Floppy Disk Interface controls are generated through this logic. The internal device timing is generated from an external crystal clock.

The FD179X has two different modes of operation according to the state of $\overline{\text{DDEN}}$. When $\overline{\text{DDEN}}$ = 0 double density (MFM) is assumed. When $\overline{\text{DDEN}}$ = 1, single



FD179X BLOCK DIAGRAM

density (FM) is assumed. 1792 & 1794 are single density only.

AM Detector — The address mark detector detects ID, data and index address marks during read and write operations.

PROCESSOR INTERFACE

The interface to the processor is accomplished through the eight Data Access Lines (DAL) and associated control signals. The DAL are used to transfer Data, Status, and Control words out of, or into the FD179X. The DAL are three state buffers that are enabled as output drivers when Chip Select (CS) and Read Enable (RE) are active (low logic state) or act as input receivers when CS and Write Enable (WE) are active.

When transfer of data with the Floppy Disk Controller is required by the host processor, the device address is decoded and CS is made low. The address bits A1 and A0, combined with the signals RE during a Read operation or WE during a Write operation are interpreted as selecting the following registers:

A1	A0	READ (RE)	WRITE (WE)
0	0	Status Register	Command Register
0	1	Track Register	Track Register
1	0	Sector Register	Sector Register
1	1	Data Register	Data Register

During Direct Memory Access (DMA) types of data transfers between the Data Register of the FD179X and the processor, the Data Request (DRQ) output is used in Data Transfer control. This signal also appears as status bit 1 during Read and Write operations.

On Disk Read operations the Data Request is activated (set high) when an assembled serial input byte is transferred in parallel to the Data Register. This bit is cleared when the Data Register is read by the processor. If the Data Register is read after one or more characters are lost, by having new data transferred into the register prior to processor readout, the Lost Data bit is set in the Status Register. The Read operation continues until the end of sector is reached.

On Disk Write operations the data Request is activated when the Data Register transfers its contents to the Data

Shift Register, and requires a new data byte. It is reset when the Data Register is loaded with new data by the processor. If new data is not loaded at the time the next serial byte is required by the Floppy Disk, a byte of zeroes is written on the diskette and the Lost Data bit is set in the Status Register.

At the completion of every command an INTRQ is generated. INTRQ is reset by either reading the status register or by loading the command register with a new command. In addition, INTRQ is generated if a Force Interrupt command condition is met.

The 179X has two modes of operation according to the state of $\overline{DDEN}$ (Pin 37). When $\overline{DDEN}$ = 1, single density is selected. In either case, the CLK input (Pin 24) is at 2 MHz. However, when interfacing with the mini-floppy, the CLK input is set at 1 MHz for both single density and double density.

GENERAL DISK READ OPERATIONS

Sector lengths of 128, 256, 512 or 1024 are obtainable in either FM or MFM formats. For FM, $\overline{DDEN}$ should be placed to logical "1." For MFM formats, $\overline{DDEN}$ should be placed to a logical "0." Sector lengths are determined at format time by the fourth byte in the "ID" field.

Sector Length Table*	
Sector Length Field (hex)	Number of Bytes in Sector (decimal)
00	128
01	256
02	512
03	1024

*1795/97 may vary — see command summary.

The number of sectors per track as far as the FD179X is concerned can be from 1 to 255 sectors. The number of tracks as far as the FD179X is concerned is from 0 to 255 tracks. For IBM 3740 compatibility, sector lengths are 128 bytes with 26 sectors per track. For System 34 compatibility (MFM), sector lengths are 256 bytes/sector with 26 sectors/track; or lengths of 1024 bytes/sector with 8 sectors/track. (See Sector Length Table)

For read operations in 8" double density the FD179X requires $\overline{RAW\ READ}$ Data (Pin 27) signal which is a 200 ns pulse per flux transition and a Read clock (RCLK) signal to indicate flux transition spacings. The RCLK (Pin 26) signal is provided by some drives but if not it may be derived externally by Phase lock loops, one shots, or counter techniques. In addition, a Read Gate Signal is provided as an output (Pin 25) on 1791/92/93/94 which can be used to inform phase lock loops when to acquire synchronization. When reading from the media in FM. RG is made true when 2 bytes of zeroes are detected. The FD179X must find an address mark within the next 10 bytes; otherwise RG is reset and the search for 2 bytes of zeroes begins all over again. If an address mark is found within 10 bytes, RG remains true as long as the FD179X is deriving any useful information from the data stream. Similarly for MFM, RG is made active when 4 bytes of "00" or "FF" are detected. The FD179X must find an address mark within the next 16 bytes, otherwise RG is reset and search resumes.

During read operations (WG = 0), the $\overline{VFOE}$ (Pin 33) is provided for phase lock loop synchronization. $\overline{VFOE}$ will go active low when:

a) Both HLT and HLD are True
b) Settling Time, if programmed, has expired
c) The 179X is inspecting data off the disk

If $\overline{WF}/\overline{VFOE}$ is not used, leave open or tie to a 10K resistor to +5.

GENERAL DISK WRITE OPERATION

When writing is to take place on the diskette the Write Gate (WG) output is activated, allowing current to flow into the Read/Write head. As a precaution to erroneous writing the first data byte must be loaded into the Data Register in response to a Data Request from the FD179X before the Write Gate signal can be activated.

Writing is inhibited when the $\overline{Write\ Protect}$ input is a logic low, in which case any Write command is immediately terminated, an interrupt is generated and the Write Protect status bit is set. The Write Fault input, when activated, signifies a writing fault condition detected in disk drive electronics such as failure to detect write current flow when the Write Gate is activated. On detection of this fault the FD179X terminates the current command, and sets the Write Fault bit (bit 5) in the Status Word. The $\overline{Write\ Fault}$ input should be made inactive when the Write Gate output becomes inactive.

For write operations, the FD179X provides Write Gate (Pin 30) and Write Data (Pin 31) outputs. Write data consists of a series of 500 ns pulses in FM ($\overline{DDEN}$ = 1) and 200 ns pulses in MFM ($\overline{DDEN}$ = 0). Write Data provides the unique address marks in both formats.

Also during write, two additional signals are provided for write precompensation. These are EARLY (Pin 17) and LATE (Pin 18). EARLY is active true when the WD pulse appearing on (Pin 30) is to be written EARLY. LATE is active true when the WD pulse is to be written LATE. If both EARLY and LATE are low when the WD pulse is present, the WD pulse is to be written at nominal. Since write precompensation values vary from disk manufacturer to disk manufacturer, the actual value is determined by several one shots or delay lines which are located external to the FD179X. The write precompensation signals EARLY and LATE are valid for the duration of WD in both FM and MFM formats.

READY

Whenever a Read or Write command (Type II or III) is received the FD179X samples the Ready input. If this input is logic low the command is not executed and an interrupt is generated. All Type I commands are performed regardless of the state of the Ready input. Also, whenever a Type II or III command is received, the TG43 signal output is updated.

COMMAND DESCRIPTION

The FD179X will accept eleven commands. Command words should only be loaded in the Command Register when the Busy status bit is off (Status bit 0). The one exception is the Force Interrupt command. Whenever a command is being executed, the Busy status bit is set. When a command is completed, an interrupt is generated and the Busy status bit is reset. The Status Register indicates whether the completed command encountered an error or was fault free. For ease of discussion, commands are divided into four types. Commands and types are summarized in Table 1.

TABLE 1. COMMAND SUMMARY

A. Commands for Models: 1791, 1792, 1793, 1794 — B. Commands for Models: 1795, 1797

		A. Bits								B. Bits							
Type	Command	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
I	Restore	0	0	0	0	h	V	r_1	r_0	0	0	0	0	h	V	r_1	r_0
I	Seek	0	0	0	1	h	V	r_1	r_0	0	0	0	1	h	V	r_1	r_0
I	Step	0	0	1	T	h	V	r_1	r_0	0	0	1	T	h	V	r_1	r_0
I	Step-in	0	1	0	T	h	V	r_1	r_0	0	1	0	T	h	V	r_1	r_0
I	Step-out	0	1	1	T	h	V	r_1	r_0	0	1	1	T	h	V	r_1	r_0
II	Read Sector	1	0	0	m	S	E	C	0	1	0	0	m	L	E	U	0
II	Write Sector	1	0	1	m	S	E	C	a_0	1	0	1	m	L	E	U	a_0
III	Read Address	1	1	0	0	0	E	0	0	1	1	0	0	0	E	U	0
III	Read Track	1	1	1	0	0	E	0	0	1	1	1	0	0	E	U	0
III	Write Track	1	1	1	1	0	E	0	0	1	1	1	1	0	E	U	0
IV	Force Interrupt	1	1	0	1	I_3	I_2	I_1	I_0	1	1	0	1	I_3	I_2	I_1	I_0

FLAG SUMMARY

TABLE 2. FLAG SUMMARY

Command Type	Bit No(s)		Description
I	0, 1	r_1 r_0 = Stepping Motor Rate See Table 3 for Rate Summary	
I	2	V = Track Number Verify Flag	V = 0, No verify V = 1, Verify on destination track
I	3	h = Head Load Flag	h = 1, Load head at beginning h = 0, Unload head at beginning
I	4	T = Track Update Flag	T = 0, No update T = 1, Update track register
II	0	a_0 = Data Address Mark	a_0 = 0, FB (DAM) a_0 = 1, F8 (deleted DAM)
II	1	C = Side Compare Flag	C = 0, Disable side compare C = 1, Enable side compare
II & III	1	U = Update SSO	U = 0, Update SSO to 0 U = 1, Update SSO to 1
II & III	2	E = 15 MS Delay	E = 0, No 15 MS delay E = 1, 15 MS delay
II	3	S = Side Compare Flag	S = 0, Compare for side 0 S = 1, Compare for side 1
II	3	L = Sector Length Flag	

	LSB's Sector Length in ID Field			
	00	01	10	11
L = 0	256	512	1024	128
L = 1	128	256	512	1024

II	4	m = Multiple Record Flag	m = 0, Single record m = 1, Multiple records
IV	0-3	I_x = Interrupt Condition Flags I_0 = 1 Not Ready To Ready Transition I_1 = 1 Ready To Not Ready Transition I_2 = 1 Index Pulse I_3 = 1 Immediate Interrupt, Requires A Reset I_3-I_0 = 0 Terminate With No Interrupt (INTRQ)	

*NOTE: See Type IV Command Description for further information.

TYPE I COMMANDS

The Type I Commands include the Restore, Seek, Step, Step-In, and Step-Out commands. Each of the Type I Commands contains a rate field (r_0 r_1), which determines the stepping motor rate as defined in Table 3.

A 2 μs (MFM) or 4 μs (FM) pulse is provided as an output to the drive. For every step pulse issued, the drive moves one track location in a direction determined by the direction output. The chip will step the drive in the same direction it last stepped unless the command changes the direction.

The Direction signal is active high when stepping in and low when stepping out. The Direction signal is valid 12 μs before the first stepping pulse is generated.

The rates (shown in Table 3) can be applied to a Step-Direction Motor through the device interface.

TABLE 3. STEPPING RATES

CLK	2 MHz	2 MHz	1 MHz	1 MHz	2 MHz	1 MHz
$\overline{\text{DDEN}}$	0	1	0	1	x	x
R1 R0	$\overline{\text{TEST}}$=1	$\overline{\text{TEST}}$=1	$\overline{\text{TEST}}$=1	$\overline{\text{TEST}}$=1	$\overline{\text{TEST}}$=0	$\overline{\text{TEST}}$=0
0 0	3 ms	3 ms	6 ms	6 ms	184μs	368μs
0 1	6 ms	6 ms	12 ms	12 ms	190μs	380μs
1 0	10 ms	10 ms	20 ms	20 ms	198μs	396μs
1 1	15 ms	15 ms	30 ms	30 ms	208μs	416μs

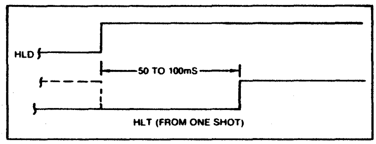
After the last directional step an additional 15 milliseconds of head settling time takes place if the Verify flag is set in Type I commands. Note that this time doubles to 30 ms for a 1 MHz clock. If $\overline{\text{TEST}}$ = 0, there is zero settling time. There is also a 15 ms head settling time if the E flag is set in any Type II or III command.

When a Seek, Step or Restore command is executed an optional verification of Read-Write head position can be performed by settling bit 2 (V = 1) in the command word to a logic 1. The verification operation begins at the end of the 15 millisecond settling time after the head is loaded against the media. The track number from the first encountered ID Field is compared against the contents of the Track Register. If the track numbers compare and the ID Field Cyclic Redundancy Check (CRC) is correct, the verify operation is complete and an INTRQ is generated with no errors. If there is a match but not a valid CRC, the CRC error status bit is set (Status bit 3), and the next encountered ID field is read from the disk for the verification operation.

The FD179X must find an ID field with correct track number and correct CRC within 5 revolutions of the media; otherwise the seek error is set and an INTRQ is generated. If V = 0, no verification is performed.

The Head Load (HLD) output controls the movement of the read/write head against the media. HLD is activated at the beginning of a Type I command if the h flag is set (h = 1), at the end of the Type I command if the verify flag (V = 1), or upon receipt of any Type II or III command. Once HLD is active it remains active until either a Type I command is received with (h = 0 and V = 0); or if the FD179X is in an idle state (non-busy) and 15 index pulses have occurred.

Head Load timing (HLT) is an input to the FD179X which is used for the head engage time. When HLT = 1, the FD179X assumes the head is completely engaged. The head engage time is typically 30 to 100 ms depending on drive. The low to high transition on HLD is typically used to fire a one shot. The output of the one shot is then used for HLT and supplied as an input to the FD179X.



HEAD LOAD TIMING

When both HLD and HLT are true, the FD179X will then read from or write to the media. The "and" of HLD and HLT appears as status Bit 5 in Type I status.

In summary for the Type I commands: if h = 0 and V = 0, HLD is reset. If h = 1 and V = 0, HLD is set at the beginning of the command and HLT is not sampled nor is there an internal 15 ms delay. If h = 0 and V = 1, HLD is set near the end of the command, an internal 15 ms occurs, and the FD179X waits for HLT to be true. If h = 1 and V = 1, HLD is set at the beginning of the command. Near the end of the command, after all the steps have been issued, an internal 15 ms delay occurs and the FD179X then waits for HLT to occur.

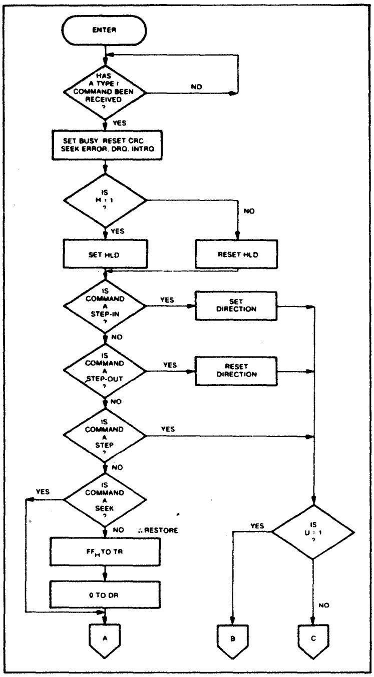
For Type II and III commands with E flag off, HLD is made active and HLT is sampled until true. With E flag on, HLD is made active, an internal 15 ms delay occurs and then HLT is sampled until true.

RESTORE (SEEK TRACK 0)

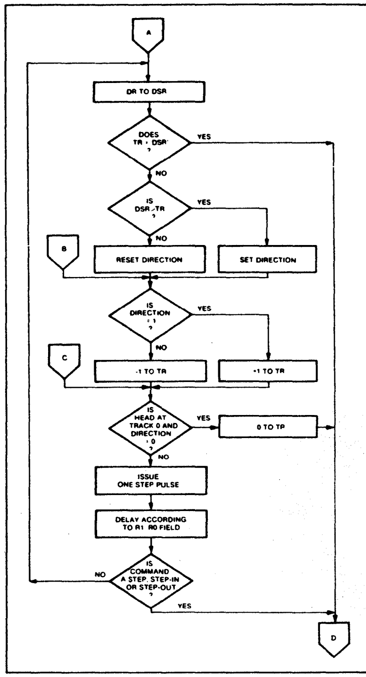
Upon receipt of this command the Track 00 ($\overline{\text{TR00}}$) input is sampled. If $\overline{\text{TR00}}$ is active low indicating the Read-Write head is positioned over track 0, the Track Register is loaded with zeroes and an interrupt is generated. If $\overline{\text{TR00}}$ is not active low, stepping pulses (pins 15 to 16) at a rate specified by the r_1 r_0 field are issued until the $\overline{\text{TR00}}$ input is activated. At this time the Track Register is loaded with zeroes and an interrupt is generated. If the $\overline{\text{TR00}}$ input does not go active low after 255 stepping pulses, the FD179X terminates operation, interrupts, and sets the Seek error status bit, providing the V flag is set. A verification operation also takes place if the V flag is set. The h bit allows the head to be loaded at the start of command. Note that the Restore command is executed when $\overline{\text{MR}}$ goes from an active to an inactive state and that the DRQ pin stays low.

SEEK

This command assumes that the Track Register contains the track number of the current position of the Read-Write head and the Data Register contains the desired track number. The FD179X will update the Track register and issue stepping pulses in the appropriate direction until the contents of the Track register are equal to the contents of



TYPE I COMMAND FLOW



TYPE I COMMAND FLOW

the Data Register (the desired track location). A verification operation takes place if the V flag is on. The h bit allows the head to be loaded at the start of the command. An interrupt is generated at the completion of the command. Note: When using multiple drives, the track register must be updated for the drive selected before seeks are issued.

STEP

Upon receipt of this command, the FD179X issues one stepping pulse to the disk drive. The stepping motor direction is the same as in the previous step command. After a delay determined by the r_1r_0 field, a verification takes place if the V flag is on. If the U flag is on, the Track Register is updated. The h bit allows the head to be loaded at the start of the command. An interrupt is generated at the completion of the command.

STEP-IN

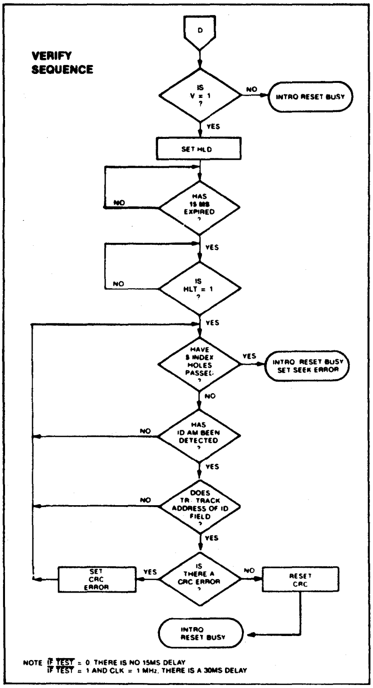
Upon receipt of this command, the FD179X issues one stepping pulse in the direction towards track 76. If the U flag is on, the Track Register is incremented by one. After a delay determined by the r_1r_0 field, a verification takes place if the V flag is on. The h bit allows the head to be loaded at the start of the command. An interrupt is generated at the completion of the command.

STEP-OUT

Upon receipt of this command, the FD179X issues one stepping pulse in the direction towards track 0. If the U flag is on, the Track Register is decremented by one. After a delay determined by the r_1r_0 field, a verification takes place if the V flag is on. The h bit allows the head to be loaded at the start of the command. An interrupt is generated at the completion of the command.

EXCEPTIONS

On the 1795/7 devices, the SSO output is not affected during Type 1 commands, and an internal side compare does not take place when the (V) Verify Flag is on.

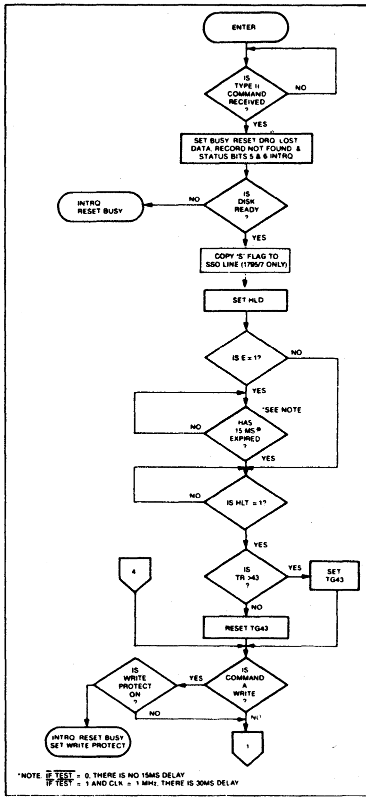


TYPE I COMMAND FLOW

TYPE II COMMANDS

The Type II Commands are the Read Sector and Write Sector commands. Prior to loading the Type II Command into the Command Register, the computer must load the Sector Register with the desired sector number. Upon receipt of the Type II command, the busy status Bit is set. If the E flag = 1 (this is the normal case) HLD is made active and HLT is sampled after a 15 msec delay. If the E flag is 0, the head is loaded and HLT sampled with no 15 msec delay. The ID field and Data Field format are shown on page 13.

When an ID field is located on the disk, the FD179X compares the Track Number on the ID field with the Track Register. If there is not a match, the next encountered ID field is read and a comparison is again made. If there was a match, the Sector Number of the ID field is compared with the Sector Register. If there is not a Sector match, the next encountered ID field is read off the disk and comparisons again made. If the ID field CRC is correct, the data field is then located and will be either written into, or read from depending upon the command. The FD179X must find an ID field with a Track number, Sector number, side number, and CRC within four revolutions of the disk; otherwise, the Record not found status bit is set (Status bit 3) and the command is terminated with an interrupt.



TYPE II COMMAND

Each of the Type II Commands contains an (m) flag which determines if multiple records (sectors) are to be read or written, depending upon the command. If m = 0, a single sector is read or written and an interrupt is generated at the completion of the command. If m = 1, multiple records are read or written with the sector register internally updated so that an address verification can occur on the next record. The FD179X will continue to read or write multiple records and update the sector register in numerical ascending sequence until the sector register exceeds the number of sectors on the track or until the Force Interrupt command is loaded into the Command Register, which terminates the command and generates an interrupt.

For example: If the FD179X is instructed to read sector 27 and there are only 26 on the track, the sector register exceeds the number available. The FD179X will search for 5 disk revolutions, interrupt out, reset busy, and set the record not found status bit.

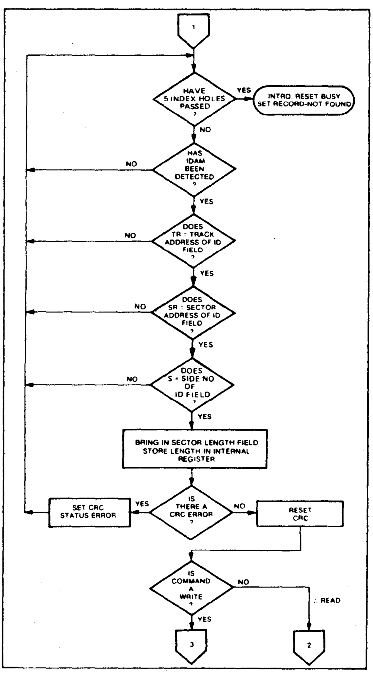
The Type II commands for 1791-94 also contain side select compare flags. When C = 0 (Bit 1) no side comparison is made. When C = 1, the LSB of the side number is read off the ID Field of the disk and compared with the contents of the (S) flag (Bit 3). If the S flag compares with the side number recorded in the ID field, the FD179X continues with the ID search. If a comparison is not made within 5 index pulses, the interrupt line is made active and the Record-Not-Found status bit is set.

The Type II and III commands for the 1795-97 contain a side select flag (Bit 1). When U = 0, SSO is updated to 0. Similarly, U = 1 updates SSO to 1. The chip compares the SSO to the ID field. If they do not compare within 5 revolutions the interrupt line is made active and the RNF status bit is set.

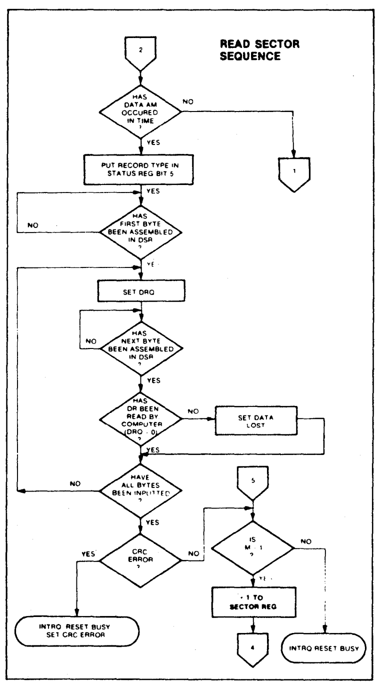
The 1795/7 READ SECTOR and WRITE SECTOR commands include a 'L' flag. The 'L' flag, in conjunction with the sector length byte of the ID Field, allows different byte lengths to be implemented in each sector. 'For IBM compatability, the 'L' flag should be set to a one.

READ SECTOR

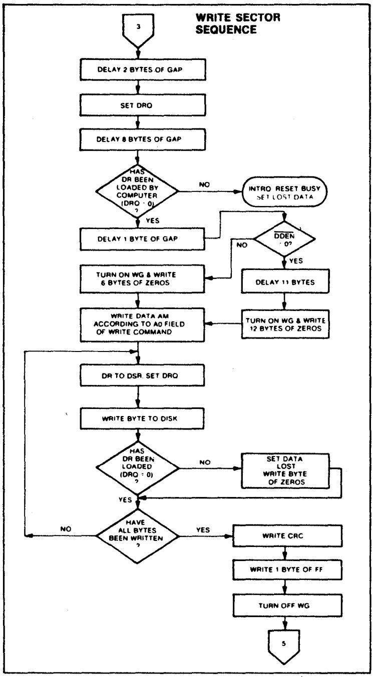
Upon receipt of the Read Sector command, the head is loaded, the Busy status bit set, and when an ID field is encountered that has the correct track number, correct sector number, correct side number, and correct CRC, the data field is presented to the computer. The Data Address



TYPE II COMMAND



TYPE II COMMAND



TYPE II COMMAND

Mark of the data field must be found within 30 bytes in single density and 43 bytes in double density of the last ID field CRC byte; if not, the ID field is searched for and verified again followed by the Data Address Mark search. If after 5 revolutions the DAM cannot be found, the Record Not Found status bit is set and the operation is terminated.

When the first character or byte of the data field has been shifted through the DSR, it is transferred to the DR, and DRQ is generated. When the next byte is accumulated in the DSR, it is transferred to the DR and another DRQ is generated. If the Computer has not read the previous contents of the DR before a new character is transferred that character is lost and the Lost Data Status bit is set. This sequence continues until the complete data field has been inputted to the computer. If there is a CRC error at the end of the data field, the CRC error status bit is set, and the command is terminated (even if it is a multiple record command).

At the end of the Read operation, the type of Data Address Mark encountered in the data field is recorded in the Status Register (Bit 5) as shown:

STATUS BIT 5	
1	Deleted Data Mark
0	Data Mark

WRITE SECTOR

Upon receipt of the Write Sector command, the head is loaded (HLD active) and the Busy status bit is set. When an ID field is encountered that has the correct track number, correct sector number, correct side number, and correct CRC, a DRQ is generated. The FD179X counts off 11 bytes in single density and 22 bytes in double density from the CRC field and the Write Gate (WG) output is made active if the DRQ is serviced (i.e., the DR has been loaded by the computer). If DRQ has not been serviced, the command is terminated and the Lost Data status bit is set. If the DRQ has been serviced, the WG is made active and six bytes of zeroes in single density and 12 bytes in double density are then written on the disk. At this time the Data Address Mark is then written on the disk as determined by the a_0 field of the command as shown below:

a_0	Data Address Mark (Bit 0)
1	Deleted Data Mark
0	Data Mark

The FD179X then writes the data field and generates DRQ's to the computer. If the DRQ is not serviced in time for continuous writing the Lost Data Status Bit is set and a byte of zeroes is written on the disk. The command is not terminated. After the last data byte has been written on the disk, the two-byte CRC is computed internally and written on the disk followed by one byte of logic ones in FM or in MFM. The WG output is then deactivated. For a 2 MHz clock the INTRQ will set 8 to 12 μsec after the last CRC byte is written. For partial sector writing, the proper method is to write the data and fill the balance with zeroes. By letting the chip fill the zeroes, errors may be masked by the lost data status and improper CRC Bytes.

TYPE III COMMANDS

READ ADDRESS

Upon receipt of the Read Address command, the head is loaded and the Busy Status Bit is set. The next encountered ID field is then read in from the disk, and the six data bytes of the ID field are assembled and transferred to the DR, and a DRQ is generated for each byte. The six bytes of the ID field are shown below:

TRACK ADDR	SIDE NUMBER	SECTOR ADDRESS	SECTOR LENGTH	CRC 1	CRC 2
1	2	3	4	5	6

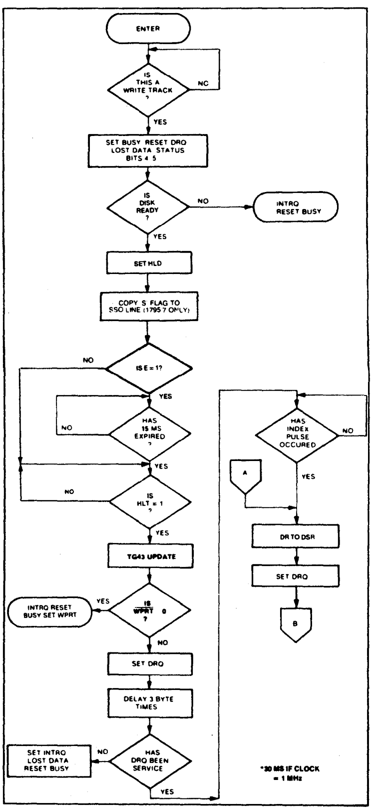
Although the CRC characters are transferred to the computer, the FD179X checks for validity and the CRC error status bit is set if there is a CRC error. The Track Address of the ID field is written into the sector register so that a comparison can be made by the user. At the end of the operation an interrupt is generated and the Busy Status is reset.

READ TRACK

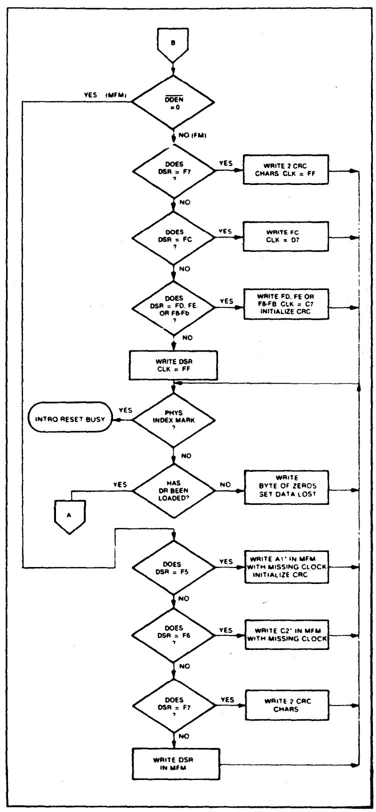
Upon receipt of the READ track command, the head is loaded, and the Busy Status bit is set. Reading starts with the leading edge of the first encountered index pulse and continues until the next index pulse. All Gap, Header, and data bytes are assembled and transferred to the data register and DRQ's are generated for each byte. The accumulation of bytes is synchronized to each address mark encountered. An interrupt is generated at the completion of the command.

This command has several characteristics which make it suitable for diagnostic purposes. They are: the Read Gate is not activated during the command; no CRC checking is performed; gap information is included in the data stream; the internal side compare is not performed; and the address mark detector is on for the duration of the command. Because the A.M. detector is always on, write splices or noise may cause the chip to look for an A.M. If an address mark does not appear on schedule the Lost Data status flag is set.

The ID A.M., ID field, ID CRC bytes, DAM, Data, and Data CRC Bytes for each sector will be correct. The Gap Bytes may be read incorrectly during write-splice time because of synchronization.



TYPE III COMMAND WRITE TRACK



TYPE III COMMAND WRITE TRACK

CONTROL BYTES FOR INITIALIZATION

DATA PATTERN IN DR (HEX)	FD179X INTERPRETATION IN FM ($\overline{DDEN}$ = 1)	FD1791/3 INTERPRETATION IN MFM ($\overline{DDEN}$ = 0)
00 thru F4	Write 00 thru F4 with CLK = FF	Write 00 thru F4, in MFM
F5	Not Allowed	Write A1* in MFM, Preset CRC
F6	Not Allowed	Write C2** in MFM
F7	Generate 2 CRC bytes	Generate 2 CRC bytes
F8 thru FB	Write F8 thru FB, Clk = C7, Preset CRC	Write F8 thru FB, in MFM
FC	Write FC with Clk = D7	Write FC in MFM
FD	Write FD with Clk = FF	Write FD in MFM
FE	Write FE, Clk = C7, Preset CRC	Write FE in MFM
FF	Write FF with Clk = FF	Write FF in MFM

*Missing clock transition between bits 4 and 5

**Missing clock transition between bits 3 & 4

WRITE TRACK FORMATTING THE DISK

(Refer to section on Type III commands for flow diagrams.)

Formatting the disk is a relatively simple task when operating programmed I/O or when operating under DMA with a large amount of memory. Data and gap information must be provided at the computer interface. Formatting the disk is accomplished by positioning the R/W head over the desired track number and issuing the Write Track command.

Upon receipt of the Write Track command, the head is loaded and the Busy Status bit is set. Writing starts with the leading edge of the first encountered index pulse and continues until the next index pulse, at which time the interrupt is activated. The Data Request is activated immediately upon receiving the command, but writing will not start until after the first byte has been loaded into the Data Register. If the DR has not been loaded by the time the index pulse is encountered the operation is terminated making the device Not Busy, the Lost Data Status Bit is set, and the Interrupt is activated. If a byte is not present in the DR when needed, a byte of zeroes is substituted.

This sequence continues from one index mark to the next index mark. Normally, whatever data pattern appears in the data register is written on the disk with a normal clock pattern. However, if the FD179X detects a data pattern of F5 thru FE in the data register, this is interpreted as data address marks with missing clocks or CRC generation.

The CRC generator is initialized when any data byte from F8 to FE is about to be transferred from the DR to the DSR in FM or by receipt of F5 in MFM. An F7 pattern will generate two CRC characters in FM or MFM. As a consequence, the patterns F5 thru FE must not appear in the gaps, data fields, or ID fields. Also, CRC's must be generated by an F7 pattern.

Disks may be formatted in IBM 3740 or System 34 formats with sector lengths of 128, 256, 512, or 1024 bytes.

TYPE IV COMMANDS

The Forced Interrupt command is generally used to terminate a multiple sector read or write command or to insure Type I status in the status register. This command can be loaded into the command register at any time. If there is a current command under execution (busy status bit set) the command will be terminated and the busy status bit reset.

The lower four bits of the command determine the conditional interrupt as follows:

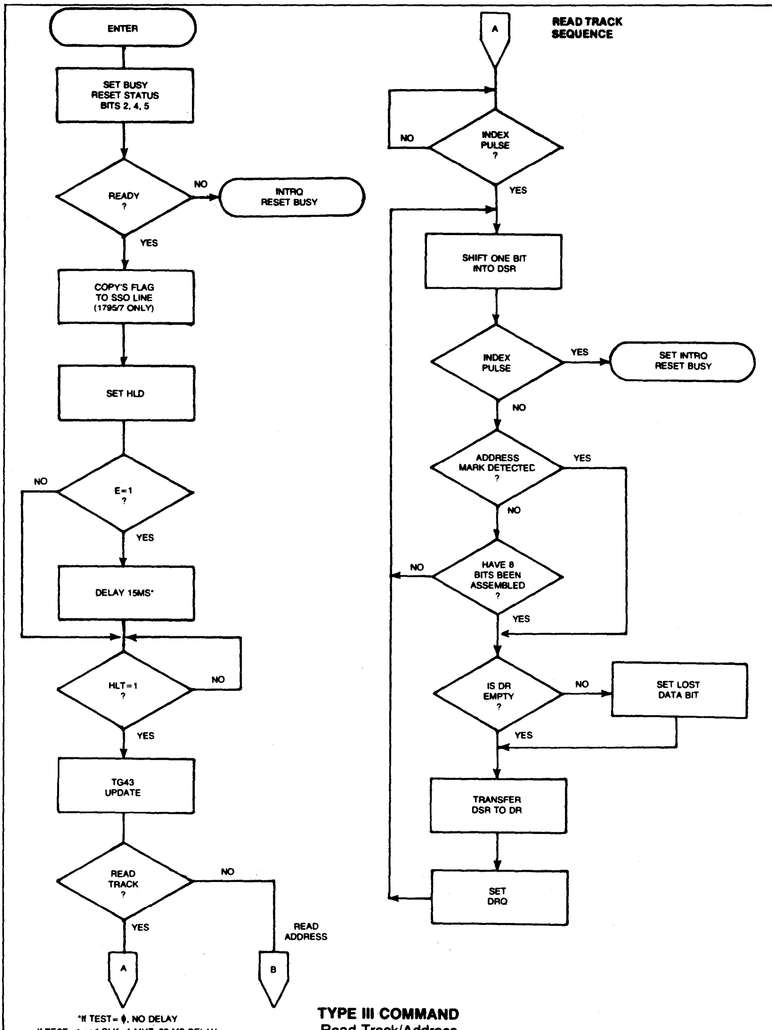
I_0 = Not-Ready to Ready Transition
I_1 = Ready to Not-Ready Transition
I_2 = Every Index Pulse
I_3 = Immediate Interrupt

The conditional interrupt is enabled when the corresponding bit positions of the command (I_3 - I_0) are set to a 1. Then, when the condition for interrupt is met, the INTRQ line will go high signifying that the condition specified has occurred. If I_3 - I_0 are all set to zero (HEX D0), no interrupt will occur but any command presently under execution will be immediately terminated. When using the immediate interrupt condition (I_3 = 1) an interrupt will be immediately generated and the current command terminated. Reading the status or writing to the command register will not automatically clear the interrupt. The HEX D0 is the only command that will enable the immediate interrupt (HEX D8) to clear on a subsequent load command register or read status register operation. Follow a HEX D8 with D0 command.

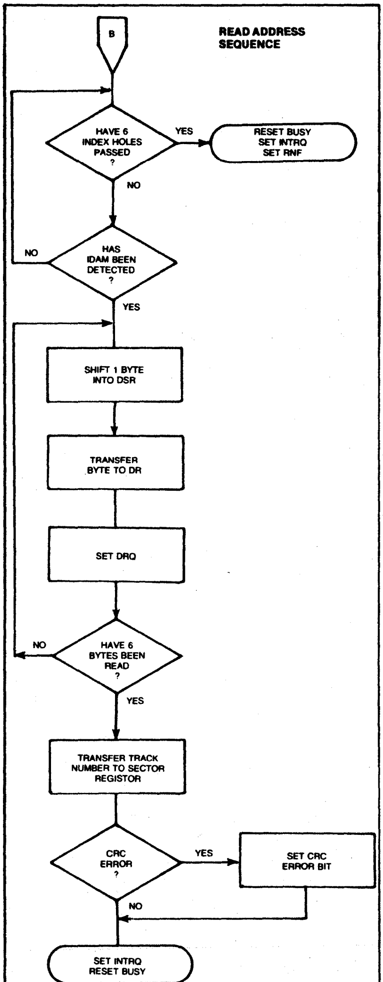
Wait 8 micro sec (double density) or 16 micro sec (single density before issuing a new command after issuing a forced interrupt (times double when clock = 1 MHz). Loading a new command sooner than this will nullify the forced interrupt.

Forced interrupt stops any command at the end of an internal micro-instruction and generates INTRQ when the specified condition is met. Forced interrupt will wait until ALU operations in progress are complete (CRC calculations, compares, etc.).

More than one condition may be set at a time. If for example, the READY TO NOT-READY condition (I_1 = 1) and the Every Index Pulse (I_2 = 1) are both set, the resultant command would be HEX "DA". The "OR" function is performed so that either a READY TO NOT- READY or the next Index Pulse will cause an interrupt condition.



TYPE III COMMAND
Read Track/Address



TYPE III COMMAND
Read Track/Address

STATUS REGISTER

Upon receipt of any command, except the Force Interrupt command, the Busy Status bit is set and the rest of the status bits are updated or cleared for the new command. If the Force Interrupt Command is received when there is a current command under execution, the Busy status bit is reset, and the rest of the status bits are unchanged. If the Force Interrupt command is received when there is not a current command under execution, the Busy Status bit is reset and the rest of the status bits are updated or cleared. In this case, Status reflects the Type I commands.

The user has the option of reading the status register through program control or using the DRQ line with DMA or interrupt methods. When the Data register is read the DRQ bit in the status register and the DRQ line are automatically reset. A write to the Data register also causes both DRQ's to reset.

The busy bit in the status may be monitored with a user program to determine when a command is complete, in lieu of using the INTRQ line. When using the INTRQ, a busy status check is not recommended because a read of the status register to determine the condition of busy will reset the INTRQ line.

The format of the Status Register is shown below:

(BITS)							
7	6	5	4	3	2	1	0
S7	S6	S5	S4	S3	S2	S1	S0

Status varies according to the type of command executed as shown in Table 4.

Because of internal sync cycles, certain time delays must be observed when operating under programmed I/O. They are: (times double when clock = 1 MHz)

Operation	Next Operation	Delay Req'd. FM	MFM
Write to Command Reg.	Read Busy Bit (Status Bit 0)	12 μs	6 μs
Write to Command Reg.	Read Status Bits 1-7	28 μs	14 μs
Write Any Register	Read From Diff. Register	0	0

IBM 3740 FORMAT — 128 BYTES/SECTOR

Shown below is the IBM single-density format with 128 bytes/sector. In order to format a diskette, the user must issue the Write Track command, and load the data register with the following values. For every byte to be written, there is one Data Request.

IBM 3740 FORMAT — 128 BYTES/SECTOR

Shown below is the IBM single-density format with 128 bytes/sector. In order to format a diskette, the user must issue the Write Track command, and load the data register with the following values. For every byte to be written, there is one Data Request.

NUMBER OF BYTES	HEX VALUE OF BYTE WRITTEN
40	FF (or 00)[1]
6	00
1	FC (Index Mark)
* 26	FF (or 00)[1]
6	00
1	FE (ID Address Mark)
1	Track Number
1	Side Number (00 or 01)
1	Sector Number (1 thru 1A)
1	00 (Sector Length)
1	F7 (2 CRC's written)
11	FF (or 00)[1]
6	00
1	FB (Data Address Mark)
128	Data (IBM uses E5)
1	F7 (2 CRC's written)
27	FF (or 00)[1]
247**	FF (or 00)[1]

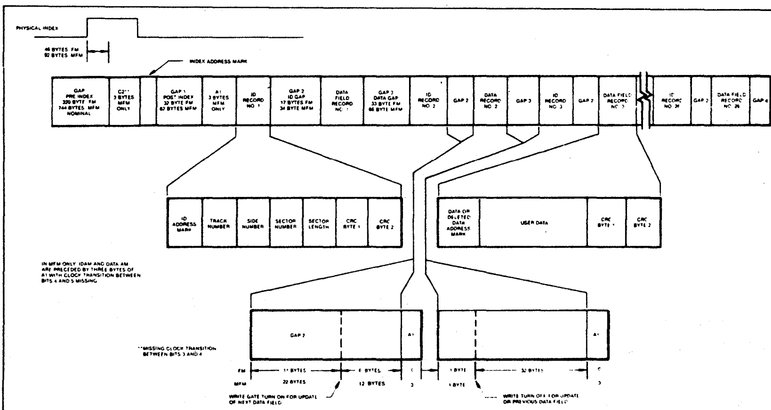
*Write bracketed field 26 times
**Continue writing until FD179X interrupts out. Approx. 247 bytes.
1-Optional '00' on 1795/7 only.

IBM SYSTEM 34 FORMAT- 256 BYTES/SECTOR

Shown below is the IBM dual-density format with 256 bytes/sector. In order to format a diskette the user must issue the Write Track command and load the data register with the following values. For every byte to be written, there is one data request.

NUMBER OF BYTES	HEX VALUE OF BYTE WRITTEN
80	4E
12	00
3	F6 (Writes C2)
1	FC (Index Mark)
* 50	4E
12	00
3	F5 (Writes A1)
1	FE (ID Address Mark)
1	Track Number (0 thru 4C)
1	Side Number (0 or 1)
1	Sector Number (1 thru 1A)
1	01 (Sector Length)
1	F7 (2 CRCs written)
22	4E
12	00
3	F5 (Writes A1)
1	FB (Data Address Mark)
256	DATA
1	F7 (2 CRCs written)
54	4E
598**	4E

*Write bracketed field 26 times
**Continue writing until FD179X interrupts out. Approx. 598 bytes.



IBM TRACK FORMAT

1. **NON-IBM FORMATS**

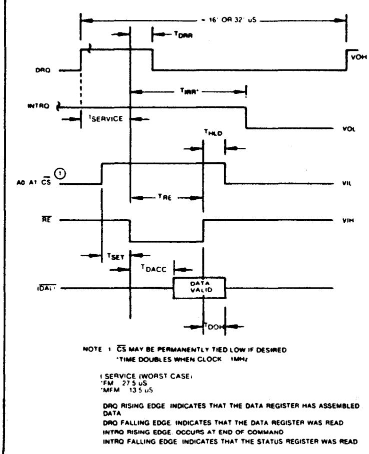
Variations in the IBM formats are possible to a limited extent if the following requirements are met:

1) Sector size must be 128, 256, 512 or 1024 bytes.
2) Gap 2 cannot be varied from the IBM format.
3) 3 bytes of A1 must be used in MFM.

In addition, the Index Address Mark is not required for operation by the FD179X. Gap 1, 3, and 4 lengths can be as short as 2 bytes for FD179X operation, however PLL lock up time, motor speed variation, write-splice area, etc. will add more bytes to each gap to achieve proper operation. It is recommended that the IBM format be used for highest system reliability.

	FM	MFM
Gap I	16 bytes FF	32 bytes 4E
Gap II	11 bytes FF	22 bytes 4E
*	6 bytes 00	12 bytes 00
*		3 bytes A1
Gap III**	10 bytes FF 4 bytes 00	24 bytes 4E 8 bytes 00 3 bytes A1
Gap IV	16 bytes FF	16 bytes 4E

*Byte counts must be exact.
**Byte counts are minimum, except exactly 3 bytes of A1 must be written.



READ ENABLE TIMING

TIMING CHARACTERISTICS

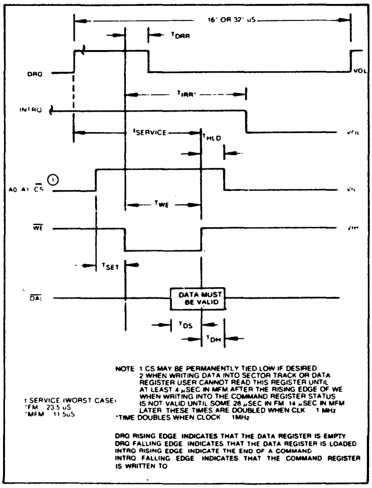
T_A = 0°C to 70°C, V_{DD} = + 12V ± .6V, V_{SS} = 0V, V_{CC} = +5V ± .25V

READ ENABLE TIMING (See Note 6, Page 21)

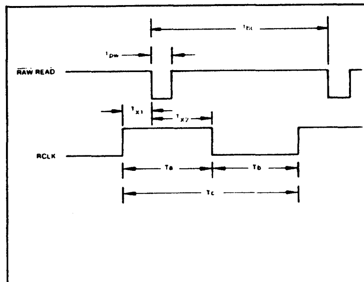
SYMBOL	CHARACTERISTIC	MIN.	TYP.	MAX.	UNITS	CONDITIONS
TSET	Setup ADDR & CS to $\overline{RE}$	50			nsec	
THLD	Hold ADDR & CS from $\overline{RE}$	10			nsec	
TRE	$\overline{RE}$ Pulse Width	400			nsec	C_L = 50 pf
TDRR	DRQ Reset from $\overline{RE}$		400	500	nsec	
TIRR	INTRQ Reset from $\overline{RE}$		500	3000	nsec	See Note 5
TDACC	Data Access from $\overline{RE}$			350	nsec	C_L = 50 pf
TDOH	Data Hold From $\overline{RE}$	50		150	nsec	C_L = 50 pf

WRITE ENABLE TIMING (See Note 6, Page 21)

SYMBOL	CHARACTERISTIC	MIN.	TYP.	MAX.	UNITS	CONDITIONS
TSET	Setup ADDR & CS to $\overline{WE}$	50			nsec	
THLD	Hold ADDR & CS from $\overline{WE}$	10			nsec	
TWE	$\overline{WE}$ Pulse Width	350			nsec	
TDRR	DRQ Reset from $\overline{WE}$		400	500	nsec	
TIRR	INTRQ Reset from $\overline{WE}$		500	3000	nsec	See Note 5
TDS	Data Setup to $\overline{WE}$	250			nsec	
TDH	Data Hold from $\overline{WE}$	70			nsec	



WRITE ENABLE TIMING



				NOMINAL		
DISKETTE	MODE	DDEN	CLK	T_a	T_b	T_c
8"	MFM	0	2 MHz	1 μs	1 μs	2 μs
8"	FM	1	2 MHz	2 μs	2μs	4 μs
5"	MFM	0	1 MHz	2 μs	2μs	4 μs
5"	FM	1	1 MHz	4 μs	4 μs	8 μs

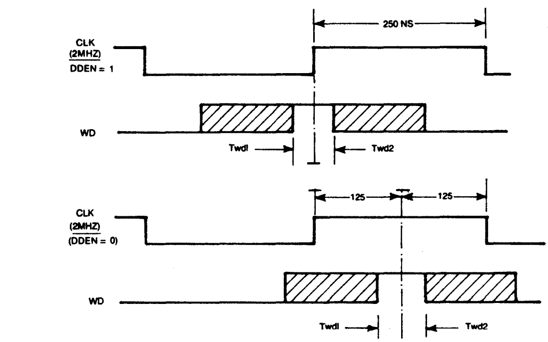
INPUT DATA TIMING

INPUT DATA TIMING:

SYMBOL	CHARACTERISTIC	MIN.	TYP.	MAX.	UNITS	CONDITIONS
Tpw	Raw Read Pulse Width	100	200		nsec	See Note 1
tbc	Raw Read Cycle Time	1500	2000		nsec	1800 ns @ 70°C
Tc	RCLK Cycle Time	1500	2000		nsec	1800 ns @ 70°C
Tx1	RCLK hold to Raw Read	40			nsec	See Note 1
Tx2	Raw Read hold to RCLK	40			nsec	See Note 1

WRITE DATA TIMING: (ALL TIMES DOUBLE WHEN CLK = 1 MHz) (See Note 6, Page 21)

SYMBOL	CHARACTERISTICS	MIN.	TYP.	MAX.	UNITS	CONDITIONS
Twp	Write Data Pulse Width		500	650	nsec	FM
			200	350	nsec	MFM
Twg	Write Gate to Write Data		2		μsec	FM
			1		μsec	MFM
Tbc	Write data cycle Time		2,3, or 4		μsec	±CLK Error
Ts	Early (Late) to Write Data	125			nsec	MFM
Th	Early (Late) From Write Data	125			nsec	MFM
Twf	Write Gate off from WD		2		μsec	FM
			1		μsec	MFM
Twdl	WD Valid to Clk	100			nsec	CLK=1 MHZ
		50			nsec	CLK=2 MHZ
Twd2	WD Valid after CLK	100			nsec	CLK=1 MHZ
		30			nsec	CLK=2 MHZ



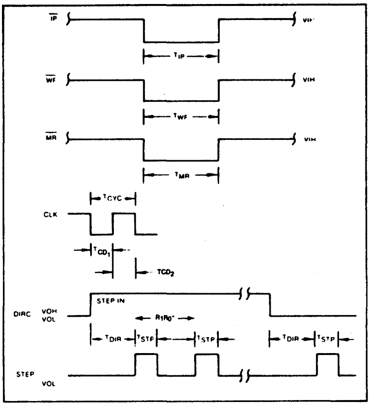
WD MUST HAVE RISING EDGE IN FIRST SHADED AREA AND TRAILING EDGE IN SECOND SHADED AREA.

WRITE DATA/CLOCK RELATIONSHIP

WRITE DATA TIMING

MISCELLANEOUS TIMING: (Times Double When Clock = 1 MHz) (See Note 6, Page 21)

SYMBOL	CHARACTERISTIC	MIN.	TYP.	MAX.	UNITS	CONDITIONS
TCD_1	Clock Duty (low)	230	250	20000	nsec	
TCD_2	Clock Duty (high)	200	250	20000	nsec	
TSTP	Step Pulse Output	2 or 4			μsec	See Note 5
TDIR	Dir Setup to Step		12		μsec	± CLK ERROR
TMR	Master Reset Pulse Width	50			μsec	
TIP	Index Pulse Width	10			μsec	See Note 5
TWF	Write Fault Pulse Width	10			μsec	



MISCELLANEOUS TIMING

*FROM STEP RATE TABLE

NOTES:

1. Pulse width on RAW READ (Pin 27) is normally 100-300 ns. However, pulse may be any width if pulse is entirely within window. If pulse occurs in both windows, then pulse width must be less than 300 ns for MFM at CLK = 2 MHz and 600 ns for FM at 2 MHz. Times double for 1 MHz.
2. A PPL Data Separator is recommended for 8" MFM.
3. tbc should be 2 μs, nominal in MFM and 4 μs nominal in FM. Times double when CLK = 1 MHz.
4. RCLK may be high or low during RAW READ (Polarity is unimportant).
5. Times double when clock = 1 MHz.
6. Output timing readings are at V_{OL} = 0.8v and V_{OH} = 2.0v.

Table 4. 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 PULSE	DRQ	DRQ	DRQ	DRQ	DRQ
S0	BUSY	BUSY	BUSY	BUSY	BUSY	BUSY

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 WRPT input.
S5 HEAD LOADED	When set, it 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 TROO input.
S1 INDEX	When set, indicates index mark detected from drive. This bit is an inverted copy of the IP input.
S0 BUSY	When set command is in progress. When reset no command is in progress.

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 RECORD 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.

ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings

V_{DD} with repect to V_{SS} (ground): +15 to −0.3V

Voltage to any input with respect to V_{SS} = +15 to −0.3V

I_{CC} = 60 MA (35 MA nominal)

I_{DD} = 15 MA (10 MA nominal)

C_{IN} & C_{OUT} = 15 pF max with all pins grounded except one under test.

Operating temperature = 0°C to 70°C

Storage temperature = −55°C to +125°C

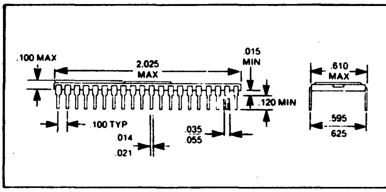
OPERATING CHARACTERISTICS (DC)

TA = 0°C to 70°C, V_{DD} = +12V ± .6V, V_{SS} = 0V, V_{CC} = +5V ± .25V

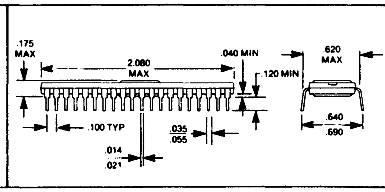
SYMBOL	CHARACTERISTIC	MIN.	MAX.	UNITS	CONDITIONS
I_{IL}	Input Leakage		10	μA	$V_{IN} = V_{DD}$**
I_{OL}	Output Leakage		10	μA	$V_{OUT} = V_{DD}$
V_{IH}	Input High Voltage	2.6		V	
V_{IL}	Input Low Voltage		0.8	V	
V_{OH}	Output High Voltage	2.8		V	I_O = −100 μA
V_{OL}	Output Low Voltage		0.45	V	I_O = 1.6 mA*
P_D	Power Dissipation		0.6	W	

*1792 and 1794 I_O = 1.0 mA

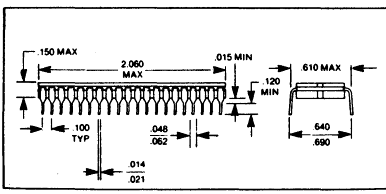
**Leakage conditions are for input pins without internal pull-up resistors. Pins 22, 23, 33, 36, and 37 have pull-up resistors. See Tech Memo #115 for testing procedures.



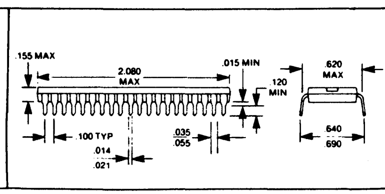
40 LEAD CERAMIC "A" or "AL"



40 LEAD RELPACK "B" or "BL"



40 LEAD CERDIP "CL"



40 LEAD PLASTIC "P" or "PL"

Notes

APPENDIX A

CP/M-80 2.2 BIOS Programming Considerations

BIOS provides the operations necessary to access the disk drives and to interface with peripherals. The user interface with the BIOS is through a series of Entry Points. These entry points are "Jump Vectors". Each jump address corresponds to a particular subroutine which performs a specific function. The Base (+ B for the jump vectors) depends on the size of RAM memory.

BIOS Entry Vector Table

BIOS Cold Boot

Entry Point:	(Bbase + 00) - Bios
Function(s):	This entry is called only by the Boot Loader to initialize CP/M.
Argument(s):	None
Value(s) Returned:	None
Registers Saved:	None
Errors Returned:	None

BIOS Warm Boot

Entry Point:	(Bbase + 03) - Bwboot
Function(s):	Perform a Warm Start by reloading the CCP and BDOS from the disk in the A: drive, returning control to the CCP.
Argument(s):	None
Value(s) Returned:	None
Registers Saved:	None
Errors Returned:	None in registers; however, message 'Boot Err' is displayed.

BIOS Console Status
Entry Point: (Bbase + 06) - Bconst

BIOS Console Input
Entry Point: (Bbase + 09) - Bconin

BIOS Console Output
Entry point: (Bbase + 0C) - Bconot

BIOS List Output
Entry Point: (Bbase + 0F) - Bprint

BIOS Punch Output
Entry Point: (Bbase + 12) - Bpunch

BIOS Reader Input
Entry Point: (Bbase + 15) - Breadr

BIOS Home Disk

Entry Point:	(Bbase + 18) - Bhome
Function(s):	Sets track number to zero in preparation for disk access.
Arguments:	None
Value(s) Returned:	None
Registers Saved:	None
Errors Returned:	None

BIOS Select Disk

Entry Point:	(Bbase + 1B) - Bseld
Function(s):	Select the requested logical disk. The drive that will be logged on in further operations is the default drive (or drive A if the default drive cannot be selected).
Arguments:	(C) = drive to select (00 - 0F)
	(E) = even if media identification required
	(E) = odd if media identification previously issued and no disks removed/replaced
Value(s) Returned:	(HL) = address of CP/M-compatible Disk Parameter Header if select successful
	(HL) = 0 otherwise
Registers Saved:	None
Errors Returned:	None

BIOS Set Track

Entry Point:	(Bbase + 1E) - Bsett
Function(s):	Stores desired track number in preparation for a disk read or write record call.
Argument(s):	(BC) = track number
Value(s) Returned:	None
Registers Saved:	None
Errors Returned:	None

BIOS Set Sector

Entry Point	(Bbase + 21) - Bsets
Function(s):	Stores desired sector number in preparation for a read or write record call.
Argument(s):	(BC) = sector number
Value(s) Returned:	None
Registers Saved:	None
Errors Returned:	None

BIOS Set DMA Address

Entry Point:	(Bbase + 24) - Bsetd
Function(s):	Stores desired transfer address in preparation for a read or write a record call.
Argument(s):	(BC) = transfer address
Value(s) Returned:	None
Registers Saved:	None
Errors Returned:	None

BIOS Read Sector

Entry Point:	(Bbase + 27) - Bread
Function(s)	Transfer one 128 (decimal) byte record from the selected disk to the current DMA transfer address.
Argument(s):	Bseld, Bsett, Bsctrn, Bsets, Bsetd previously called.
Value(s) Returned:	None
Registers Saved:	none
Errors Returned:	(A) = 00 if no error (A) = FF if error

BIOS Write Sector

Entry Point:	(Bbase + 2A) - Bwritt
Function(s):	Transfer one 128 (decimal) byte record from the current DMA transfer address to the selected disk.
Argument(s):	Bseld, Bsett, Bsctrn, Bsets, Bsetd previously called.
Value(s) Returned:	None
Registers Saved:	None
Errors Returned:	(A) = 00 if no error (A) = FF if error

BIOS List Status

Entry Point:	(Bbase + 2D) - Bprnts

BIOS Sector Translate

Entry Point:	(Bbase + 30) - Bsctrn		
Function(s):	Translate a logical sector number into a physical sector number in preparation for a call to Bsets, the BIOS set sector call.		
Argument(s):	(BC)	=	Sector number (0 < = (BC) < sectors per track
	(DE)	=	Skew table address obtained from the CP/M Disk Parameter Header
Value(s) Returned:	(HL)	=	(BC) if (DE) = 0
	(L)	=	[(DE) + (BC)] if (DE) = 0
	(H)	=	(B) should be 0
Registers Saved:	None		
Errors Returned:	None		

Notes

APPENDIX B

Monitor Entry Vector Table

F000H	Cold start monitor
F003H	Warm start monitor
F006H	Keyboard status
F009H	Keyboard input
F00CH	CRT output
F00FH	Fast CRT output from C
F012H	SIO channel B input status
F015H	SIO channel B input
F018H	SIO channel B output
F01BH	Drive select
F01EH	Home r/w head
F021H	Seek to track
F024H	Read sector
F027H	Write sector
F02AH	Execute physical driver request
F02DH	Set direct CRT cursor
F030H	Direct CRT display
F033H	CRT memory block move
F036H	Return address of disk mapping table
F039H	Return address of day variable
F03CH	Return configuration status
F03FH	SIO channel B output ready status
F042H	Set configuration
F045H	Start screen print
F048H	Accessible 1-second interrupt
F04BH	Console status through iobyte
F04EH	Console input through iobyte
F051H	Console output through iobyte
F054H	Printer output through iobyte
F057H	Printer status through iobyte
F05AH	Communications input ready status
F05DH	Communications input data
F060H	Communications output data
F063H	Communications output ready status
F066H	Idle while i/o is pending
F069H	Record soft error

Notes

APPENDIX C

Documented System Storage and Structures

Z80-A Mode 2 Interrupt Vectors

```
FF00   SIOVO:   DEFS2    ;Z80-A SIO port B xmit buffer empty
FF02   SIOV1:   DEFS2    ;Z80-A SIO port B external/status change
FF04   SIOV2:   DEFS2    ;Z80-A SIO port B receive data available
FF06   SIOV3:   DEFS2    ;Z80-A SIO port B special receive
                          condition
FF08   SIOV4:   DEFS2    ;Z80-A SIO port  A xmit buffer empty
FF0A   SIOV5:   DEFS2    ;Z80-A SIO port A external/status change
FF0C   SIOV6:   DEFS2    ;Z80-A SIO port A receive data available
FF0E   SIOV7:   DEFS2    ;Z80-A SIO port A special receive
                          condition

FF10   CTCVO:   DEFS2    ;Z80-A CTC channel 0 interrupt
FF12*  CTCV1:   DEFS2    ;Z80-A CTC channel 1 interrupt
FF14   CTCV2:   DEFS2    ;Z80-A CTC channel 2 interrupt
FF16*  CTCV3:   DEFS2    ;Z80-A CTC channel 3 interrupt

FF18   SYSVA:   DEFS2    ;System Z80-A PIO port A interrupt
FF1A*  SYSVB:   DEFS2    ;System Z80-A PIO port B interrupt

FF1C   GENVA:   DEFS2    ;General purpose Z80-A PIO port A
                          interrupt
FF1E   GENVB:   DEFS2    ;General purpose Z80-A PIO port B
                          interrupt
```

*Vectors used by the Monitor ROM

Keyboard Data Input FIFO Variables

```
FF20   fifo:    defs 16  ;Console input fifo
FF30   fifcnt:  defs 1   ;FIFO data counter
FF31   fifin:   defs 1   ;FIFO input pointer
FF32   fifout:  defs 1   ;FIFO output pointer

FF33            defs 1   ;Round address
```

More Interrupt Vectors

```
FF34    expvec:   defs 8      ;Space for 4 vectors for expansion slot
```

Available Memory Pointers

```
FF3C    availb:   defs 2      ;Bottom of available memory
FF3E    availt:   defs 2      ;Top of available memory
```

End of documented storage locations

Logical to Physical Drive Mapping Tables
Seltab contains two bytes per logical CP/M drive A-P. The first byte is an index into the physical driver address table (see next table). The second byte is a unit number that is passed to the driver by the XQDVR dispatcher.

```
Seltab:

    A:   defb    1,0     ;Floppy unit 0
    B:   defb    1,1     ;Floppy unit 1
    C:   defb    1,2     ;Floppy unit 2
    D:   defb    1,3     ;Floppy unit 3

    E:   defb    1,4     ;Rigid partition 0
    F:   defb    1,5     ;Rigid partition 1
    G:   defb    1,6     ;Rigid partition 2
    H:   defb    1,7     ;Rigid partition 3
    I:   defb    0,0     ;Error driver
    J:   defb    0,0     ;Error driver
    K:   defb    0,0     ;Error driver
    L:   defb    0,0     ;Error driver
    M:   defb    0,0     ;Error driver
    N:   defb    0,0     ;Error driver
    O:   defb    0,0     ;Error driver
    P:   defb    0,0     ;Error driver
```

Physical Driver Address Table

Drvtab contains the addresses of several independent physical disk drivers. By convention, driver number 0 always returns a select error. Unused entries in Seltab should point to this trivial driver.

```
Drvtab:   defw   Selerr   ;Select error physical driver
          defw   Dskdvr   ;Disk driver (WD or SA)
          defw   0        ;Empty physical driver
                          ;Expansion slots

          defw   0
          defw   0
          defw   0
          defw   0
          defw   0-1      ;Mark last entry
```

Physical Driver Request Block

```
db    command        ;FF = Select
                     ;00 = Write
                     ;01 = Read
ds    1              ;For system use
db    Ldrive         ;Logical drive for request (00 - 0F)
dw    Track          ;Track number for request
dw    Sector         ;Sector number for request
dw    Address        ;Address of sector buffer for request
```

Time-of-Day and Timer Variables

```
        Milsec:     ds   2     ;Location incremented by CTC1
                               ;Interrupt
                    ds   2     ;(unused)
        Ticker:     ds   2     ;Increments once per second
        Steprt:     ds   1     ;WD1797 step rate
         Motor:     ds   1     ;Disk motor/select timeout (1 Hz)
HL→        Day:     ds   1     ;01-31
         Month:     ds   1     ;01-12
          Year:     ds   1     ;80-99
          Hour:     ds   1     ;00-23
        Minute:     ds   1     ;00-59
        Second:     ds   1     ;00-59
        Linbuf:     ds   80    ;Line buffer
```

How To Make Monitor Calls from Basic

Several of the monitor function calls return the value in the HL register if the H register equals 0, or return the value at the address pointed to by the HL register if the H register is not zero. This convention allows Microsoft Basic Users to access these functions directly. The examples listed in this section demonstrate this feature of the ROSR ROM.

```
100   '
110   ' Make 820-II Monitor call to get address of day variable, then
120   ' Print Day, Month etc.
130   '
140   DATA Day,Month,Year,Hour,Minute,Second
150   '
160   DEFINT I
170   GETTOD = &HF039:CALL GETTOD(I)          'Return Add. of Day
180   FOR X = 0 TO 5
190       READ X$
200       PRINT USING "\      \ ##;X$,PEEK(I + X)
210   NEXT X
220   END
```

```
100   '
110   ' Do configuration status call & print value returned
120   '
130   DEFINT I
140   GETCON = &HF03C:CALL GETCON(I)          'Get config status
150   PRINT CHR$(26);                   'Clear Screen
160   PRINT "The configuration status word is - ";
170   PRINT HEX$(I);
180   PRINT " (Hex)"
190   END
```

```
100   ' Example Using Line Delete To scroll screen up.
110   ' Make 820-II Monitor Call to get address of day variable
120   ' then calculate address of line input buffer variable.
130   '
140   ' Clear screen, fill screen with characters, position
150   ' Cursor back on top line, send line delete code to CRT,
160   ' This moves the line deleted from the top of the screen
170   ' To the input buffer.
180   '
190   ' Recall deleted line from line input buffer & display
200   ' on line 23 of the screen.
210   '
220   '
230   WIDTH 255
240   PRINT CHR$(5);" ";              'Remove cursor
250   DEFINT I
260   GETTOD = &HF039:CALL GETTOD(I) 'Get address of Day Variable
270   I = I + 6            'Line input buffer is at Day + 6
280   PRINT CHR$(26);          'Clear screen
290   FOR X = 1 TO 23
300       PRINT STRING$(80,CHR$(X + 64)); 'Fill Screen
310   NEXT X
320   '
330   FOR M = 1 TO 100                'Do 100 lines
340       PRINT CHR$(30);             'Put Cursor back on top line
350       PRINT CHR$(27);"R";         'Do line delete, move deleted
360                                   'Line to buffer.
370       PRINT  CHR$(27);" = ";CHR$(32 + 22);CHR$(32)
380       FOR X = 0 TO 79             'Now print characters back from
390             PRINT CHR$(PEEK(I + X);  'Input buffer
400       NEXT X
410   NEXT M
420   PRINT CHR$(26);CHR$(5);CHR$(2);    'Clear screen and
430   '                                  Restore Cursor.
440   END
```

```
100   ' Example Using Line Insert To scroll screen down.
110   ' Make 820-II Monitor Call to get address of day variable
120   ' then calculate address of line input buffer variable.
130   '
140   ' Clear screen, fill screen with characters, position
150   ' Cursor back on top line, send line insert code to CRT,
160   ' This moves the line deleted from the bottom of the screen
170   ' To the input buffer.
180   '
190   ' Recall deleted line from line input buffer & display
200   ' on the first line of the screen.
210   '
220   '
230   WIDTH 255
240   PRINT CHR$(5);" ";                        'Remove cursor
250   DEFINT I
260   GETTOD = &HF039:CALL GETTOD(I)  'Get address of Day Variable
270   I = I + 6               'Line input buffer is at Day + 6
280   PRINT CHR$(26);         'Clear screen
290   FOR X = 1 TO 23
300       PRINT STRING$(80,CHR$(X + 64)); 'Fill Screen
310   NEXT X
320   '
330   FOR M = 1 TO 100:       'Do 100 lines
340       PRINT CHR$(30);     'Put Cursor back on top line
350       PRINT CHR$(27);"E"; ' Do line insert, move deleted
360                           'Line to buffer.
370       PRINT  CHR$(27);" = ";CHR$(32 + 22);CHR$(32)
380       FOR X = 0 TO 79     'Now print characters back from
390           PRINT CHR$(PEEK(I + X);  'Input buffer
400       NEXT X
410   NEXT M
420   PRINT CHR$(26);CHR$(5);CHR$(2);   'Clear screen and
430   '                                  Restore Cursor.
440   END
```

Bank Switching

- The Bank control switch is bit 7 of port 1C.
  Bit 7 = 0 = Bank 1 (RAM)
  Bit 7 = 1 = Bank 0 (ROM)

- Change bit 7 only: Bits 0 through 6 should be maintained.

- Bank 0 and 1 are mutually exclusive; data movement to or from one bank will not affect the other.

- When bank switching, the driver code must be executed at C000h or above; the upper 16K (C000h-FFFFh) is common memory to both banks.

For example,

```
DI
IN A,(1Ch)        ;read port
SET 7,a           ;set bit
EI
OUT (1Ch),a       ;output

DI
IN A,(1Ch)
RES 7,a           ;reset bit
EI
OUT (1Ch)
```

```
                                title   Balcones Operating System for the XEROX 820-II
                                .z80
                        ;;;     Balcones Operating System for XEROX 820-II.
                        ;
                        ;       Copyright 1982 (C) Balcones Computer Corporation
                        ;
                        ;                       All rights reserved
                        ;
                        ;;                      Robert Burns, Bcc.
                        ;
        0191            rev     defl    401
                                subttl  Symbol Definitions
                                page
```

```

FFFF              true    equ     -1
0000              false   equ     not true

0000              debug   equ     false           ;assemble ram loader

                  ;;      Absolute Memory Addresses.
                  ;
0000              rom     equ     01000h and debug;non resident code base
1800              romsiz  equ     01000h+((not debug) and 0800h)
1800              Rx1984  equ     01800h          ;prescription for the future
0800              Lx1984  equ     00800h          ;length of future
0003              iobyte  equ     00003h          ;i/o byte
0080              bootld  equ     00080h          ;boot loader address
ED80              bootbf  equ     0ed80h          ;boot loader buffer
FF00              ram     equ     0ff00h          ;system ram page address
F000              monitr  equ     0f000h          ;resident monitor address
3000              crtmem  equ     03000h          ;crt memory address
3C00              crtmax  equ     crtmem+24*128   ;crt maximum address
0030              crtbas  equ     high crtmem     ;starting page of display ram
003C              crttop  equ     high crtmax     ;ending page of display ram

                  ;;      I/O Port Addresses.
                  ;
0000              bauda   equ     00h             ;channel a baud rate generator
0004              siodpa  equ     04h             ;sio data port A (communications)
0005              siodpb  equ     05h             ;sio data port B (printer)
0006              siocpa  equ     06h             ;sio control/status port A
0007              siocpb  equ     07h             ;sio control/status port B
0008              gpioda  equ     08h             ;general purpose parallel i/o A data
0009              gpioca  equ     09h             ;general purpose parallel i/o A control
000A              gpiodb  equ     0ah             ;general purpose parallel i/o B data
000B              gpiocb  equ     0bh             ;general purpose parallel i/o B control
000C              baudb   equ     0ch             ;channel b baud rate generator
0010              wd1797  equ     10h             ;western digital disk controller base
0014              scroll  equ     14h             ;crt bottom line scroll register
0018              ctc     equ     18h             ;quad counter/timer circuit
0018              ctc0    equ     18h             ;ctc channel 0 (user)
0019              ctc1    equ     19h             ;ctc channel 1 (msec, screen print)
001A              ctc2    equ     1ah             ;ctc channel 2 (one second prescaler)
001B              ctc3    equ     1bh             ;ctc channel 3 (one second)
001C              syspio  equ     1ch             ;system pio data
001D              sysctl  equ     1dh             ;system pio control
001E              kbddat  equ     1eh             ;keyboard data
001F              kbdctl  equ     1fh             ;keyboard control
0028              bellof  equ     28h             ;turn bell off
0029              bellon  equ     29h             ;turn bell on
0030              slsden  equ     30h             ;select single density
0031              sldden  equ     31h             ;select double density
0034              chrom1  equ     34h             ;select ROM 1 character generator
0035              chrom2  equ     35h             ;select ROM 2 character generator
0036              lowlite equ     36h             ;select low intensity attribute
0068              async   equ     68h             ;set internal clocks for asynchronous sio A
0069              sync    equ     69h             ;set external clocks for  synchronous sio A

```

```
                        ;;      Configuration Status Byte Bit Definitions.
                        ;
        0007            c.keym  equ     7               ;Keyboard upper bit is passed
        0006            c.sasi  equ     6               ;Shugart SA-1403D Disk Controller
        0004            c.five  equ     4               ;Five inch micro floppies

                        ;;      Ascii.
                        ;
        0004            eot     equ     04h             ;ascii end of text
        000A            lf      equ     0ah             ;ascii line feed
        000D            cr      equ     0dh             ;ascii carriage return
        0011            xon     equ     11h             ;ascii Xon
        0013            xoff    equ     13h             ;ascii Xoff
        001B            esc     equ     1bh             ;ascii escape
        001A            clrs    equ     1ah             ;clear screen

                        ;;      Special Key Constants.
                        ;
        001E            Helpkey equ     01eh
        009E            Scrprt  equ     09eh            ;Screen Print key CTRL <HELP>
        009B            Abort   equ     09bh            ;Automatic Abort  CTRL <ESC>

                        ;;      Bell Constants.
                        ;
        0035            bltim   equ     35h             ;bell loop time
        0061            blonc   equ     61h             ;bell on   time
        0061            blofc   equ     61h             ;bell off  time

                        ;;      Assembly Options.
                        ;
        8000            o.resv  equ     1000000000000000b       ;reserved
        4000            o.auto  equ     0100000000000000b       ;auto boot A:
        2000            o.help  equ     0010000000000000b       ;help command
        1000            o.prot  equ     0001000000000000b       ;printer protocol
        0800            o.ddvr  equ     0000100000000000b       ;disk drivers
        0400            o.baud  equ     0000010000000000b       ;baud rate set command
        0200            o.inpc  equ     0000001000000000b       ;in command
        0100            o.outc  equ     0000000100000000b       ;out command
        0080            o.verf  equ     0000000010000000b       ;verify memory block
        0040            o.ramt  equ     0000000001000000b       ;simple ram test
        0020            o.disk  equ     0000000000100000b       ;console disk read/write commands
        0010            o.esct  equ     0000000000010000b       ;escape command table
        0008            o.type  equ     0000000000001000b       ;typewriter mode
        0004            o.fill  equ     0000000000000100b       ;fill memory
        0002            o.move  equ     0000000000000010b       ;move memory
        0001            o.term  equ     0000000000000001b       ;terminal scroll driver

        0000            options defl    debug    and not o.ddvr and not o.esct
        0000            options defl    options  and not o.disk and not o.resv
        0000            options defl    options  and not o.verf and not o.fill
        0000            options defl    options  and not o.ramt

        BFFF            options defl    (not debug or o.esct) and not o.auto

                        ;;      configuration sector offsets.
                        ;
```

```
EE5F        z.scra  equ     bootbf+255-32   ;initial screen attribute
EE60        z.stpr  equ     z.scra+1        ;floppy step rate
EE62        z.keym  equ     z.stpr+2        ;keyboard mask
EE63        z.sioA  equ     z.keym+1        ;sio A init
EE6D        z.sioB  equ     z.sioA+10       ;sio B init
EE77        z.siom  equ     z.sioB+10       ;clear to send low/high/ignore
EE79        z.siov  equ     z.siom+2        ;data carrier detect low/high/ignore
EE7B        z.xonp  equ     z.siov+2        ;Xon/Xoff protocal
EE7D        z.baua  equ     z.xonp+2        ;comm channel baud rate
EE7E        z.baub  equ     z.baua+1        ;printer baud rate
EE7F        z.iobt  equ     z.baub+1        ;initial i/o byte

            ;;      parallel printer status bits.
            ;
0007        p.ackn  equ     7               ;acknowledge
0006        p.onln  equ     6               ;on line
0005        p.rdyi  equ     5               ;ready to input
0004        p.rdyo  equ     4               ;ready to output
0002        p.strb  equ     2               ;data stobe
0000        p.auto  equ     0               ;auto LF enable

                    subttl  Code Generation Control Macros Definitions
                    page
```

```

        ;;      Rom code placement macros.
        ;
        ;;      The Common Segment holds the non-resident (banked) portion
        ;       of the monitor.  This segment is not copied to ram.
        ;
        ;;      The Data Segment holds the resident portion of the monitor.
        ;       It is moved to ram at location MONITR during initialization.
        ;
        ;;      The Code Segment holds the various Transient Commands.  Each
        ;       command is loaded from the ROM to the TPA when it is executed.
        ;
        ;;      The following macros keep it all straight.
        ;
        ;;      below - Generate code for rom below.
        ;
below   macro
        segment b               ;;enable common segment
        endm

        ;;      above - Generate code for ram above.
        ;
above   macro
        segment d               ;;enable data segment
        endm

        ;;      Overlay - Generate code for transients.
        ;
overlay macro   addr
tloc    defl    tloc+$-cloc
addr    equ     tloc+bloc+cloc-Monitr
        segment c               ;;enable code segment
        endm

        ;;      bseg - activate common segment.
        ;
bseg    macro
        common  /COMROM/
        defs    comres
sega    defl    $
        endm

        ;;      segment - Activate Segment.
        ;
segment macro   s
        update                  ;;update active phase counter
s&space defl    -1              ;;set enabled segment active
        s&seg                   ;;activate segment code placement
        .phase  s&loc           ;;set absolute segment location counter
        endm

        ;;      update - Update Phase Counters.
        ;
update  macro
        if      bspace
```

```
                        comres  defl    $-rom+100h-3
                                endif
                                irpc    x,<bcd>
                                if      x&space         ;;if segment active
                                if      cspace
                                if      tpal lt ($-cloc)
                        tpal    defl    $-cloc
                                endif
                                else
                        x&loc   defl    $               ;;save segment address
                                endif
                        x&space defl    0               ;;clear segment active
                                .dephase                ;;revert to relocatable
                                endif
                                endm
                                endm

        0000            bloc    defl    rom             ;establish non-resident code base
        F000            dloc    defl    monitr          ;establish resident code base
        0000            tloc    defl    0               ;establise Transient code base
        0000            tpal    defl    0               ;establish maximum transient length
        0000            bspace  defl    0               ;preset common segment inactive
        0000            cspace  defl    0               ;preset code segment inactive
        0000            dspace  defl    0               ;preset data segment inactive
        0000            comres  defl    0               ;preset common base address

                                subttl  Ram Loader for Testing Only
                                page
```

```

                                           bseg
    0000!                     +            defs    comres
    0000!                          entry:
 
    0000!    21 00FD!              xcks:   ld      hl,bbase+movln
    0003!    01 17FF                       ld      bc,romsiz-1
    0006!    1E 00                         ld      e,0              ;preset checksum
    0008!    7E                    xcks1:  ld      a,(hl)
    0009!    23                            inc     hl
    000A!    83                            add     a,e
    000B!    5F                            ld      e,a
    000C!    0B                            dec     bc
    000D!    78                            ld      a,b
    000E!    B1                            or      c
    000F!    20 F7                         jr      nz,xcks1
    0011!    7B                            ld      a,e              ;store twos complement of checksum
    0012!    ED 44                         neg
    0014!    77                            ld      (hl),a           ;store checksum
    0015!    C3 0000                       jp      0

                                           subttl  System Initialization
                                           page
```

```


0018!                    bbase:                         ;symbol for accessing non-resident base address
0018!                            defs    100h-($-(entry-3)),-1

00E5                     movln   equ     $-bbase
00FD                     comres  defl    100h-3

                                 below                  ;generate non-resident code
0000!              +             defs    comres

                         ;;      prs - preset storage.
                         ;
                         ;       Entry:  Power up or Reset button.
                         ;
0000    F3               prs:    di                     ;lock up system
0001    AF                       xor     a
0002    3D               prs1:   dec     a              ;the pause that refershes
0003    20 FD                    jr      nz,prs1
0005    ED 73 FFE0               ld      (rstsp),sp     ;save partial reset state
0009    22 FFE2                  ld      (rsthl),hl     ;in case the luser go boom
000C    E1                       pop     hl             ;pick possible return off stack
000D    22 FFE4                  ld      (rstpc),hl
0010    D9                       exx                    ;give primary registers half a break
0011    1C                       inc     e
0012    31 3839                  ld      sp,3839h       ;load strange values in SP
0015    31 4142                  ld      sp,4142h
0018    4C                       ld      c,h            ;insure
0019    43                       ld      b,e            ;registers
001A    4F                       ld      c,a            ;can
001B    4E                       ld      c,(hl)         ;forget
001C    45                       ld      b,l            ;insure
001D    53                       ld      d,e            ;registers
001E    43                       ld      B,e            ;can
001F    4F                       ld      C,a            ;copy
0020    4D                       ld      C,l
0021    50                       ld      d,b
0022    55                       ld      d,l
0023    54                       ld      d,h
0024    45                       ld      b,l
0025    52                       ld      d,d
0026    08                       ex      af,af'
0027    3E 17                    ld      a,24-1         ;line up bottom of screen
0029    D3 14                    out     (scroll),a     ;init scroll port
002B    21 3000                  ld      hl,crtmem      ;clear display memory
002E    36 20                    ld      (hl),' '
0030    11 3001                  ld      de,crtmem+1
0033    01 0BFF                  ld      bc,crtmax-crtmem-1
0036    ED B0                    ldir                   ;pray the video hardware works
0038    31 F000                  ld      sp,monitr      ;insure monitor ram ok
003B    21 AA55          prs2:   ld      hl,0aa55h      ;walk checker board through ram
003E    C1                       pop     bc             ;read ram
003F    E5                       push    hl             ;write ram fast
0040    D1                       pop     de             ;read ram fast
0041    C5                       push    bc             ;put ram back
```

```
0042    F1                      pop     af              ;and verify it
0043    90                      sub     b
0044    20 76                   jr      nz,err1         ;if ram failure
0046    ED 52                   sbc     hl,de
0048    20 72                   jr      nz,err1         ;if ram or register failure
004A    3B                      dec     sp              ;advance test address
004B    3F                      ccf
004C    ED 7A                   adc     hl,sp
004E    20 EB                   jr      nz,prs2         ;if top of memory not reached
0050    31 0000                 ld      sp,stack        ;set monitor stack
0053    21 0000                 ld      hl,prs          ;set rom address
0056    01 1800                 ld      bc,romsiz
0059    CD 00AF                 call    ccs             ;compute check sum
005C    20 63                   jr      nz,err2         ;if bad rom
005E    21 00E6                 ld      hl,intab        ;point to default variable table
0061    06 00           prs3:   ld      b,0
0063    4E                      ld      c,(hl)          ;set data block length
0064    23                      inc     hl
0065    5E                      ld      e,(hl)          ;set variable address in ram
0066    23                      inc     hl
0067    56                      ld      d,(hl)
0068    23                      inc     hl              ;point to initial values
0069    ED B0                   ldir                    ;copy data from rom to variables in ram
006B    CB 7E                   bit     7,(hl)
006D    28 F2                   jr      z,prs3          ;if more data to preset
006F    23                      inc     hl              ;point to i/o init data table
0070    46              prs4:   ld      b,(hl)          ;set number of bytes to preset
0071    23                      inc     hl
0072    4E                      ld      c,(hl)          ;set i/o port address
0073    23                      inc     hl
0074    ED B3                   otir                    ;shoot preset data to i/o device
0076    CB 7E                   bit     7,(hl)
0078    28 F6                   jr      z,prs4          ;if more devices require initialization
007A    DB 1E                   in      a,(kbddat)      ;assert PARDY
007C    ED 5E                   im      2               ;select interrupt mode 2
007E    3E FF                   ld      a,high vectab   ;set interrupt vector page
0080    ED 47                   ld      i,a
0082    21 041B                 ld      hl,rbase        ;set resident base address
0085    11 F000                 ld      de,monitr       ;set monitor address
0088    01 0F00                 ld      bc,ram-monitr   ;set max resident length
008B    ED B0                   ldir                    ;plant monitor upstairs
008D    21 1800                 ld      hl,Rx1984       ;prognosticate
0090    01 0800                 ld      bc,Lx1984
0093    CD 00AF                 call    ccs
0096    20 14                   jr      nz,prs5
0098    2A 1FFD                 ld      hl,(Rx1984+Lx1984-3)
009B    11 55AA                 ld      de,55aah
009E    ED 52                   sbc     hl,de
00A0    21 FAD8                 ld      hl,cmdtab
00A3    11 F360                 ld      de,seltab
00A6    01 FC55                 ld      bc,cloc
00A9    CC 1800                 call    z,Rx1984        ;FutureShock
00AC    C3 FC55         prs5:   jp      signon          ;Signon Resident Monitor

00AF    1E 00           ccs:    ld      e,0             ;preset ckecksum
00B1    7E              ccs1:   ld      a,(hl)
```

```
00B2    23                          inc     hl
00B3    83                          add     a,e
00B4    5F                          ld      e,a
00B5    0B                          dec     bc
00B6    78                          ld      a,b
00B7    B1                          or      c
00B8    20 F7                       jr      nz,ccs1
00BA    B3                          or      e
00BB    C9                          ret

00BC    21 00D4             errl:   ld      hl,erml         ;set ram error message
00BF    18 03                       jr      err
00C1    21 00DD             err2:   ld      hl,errm2
00C4    11 3024             err:    ld      de,crtmem+40-(errml/2)
00C7    01 0009                     ld      bc,errml
00CA    ED B0                       ldir
00CC    0B                  err3:   dec     bc              ;pause a while
00CD    78                          ld      a,b
00CE    B1                          or      c
00CF    20 FB                       jr      nz,err3
00D1    C3 0000                     jp      prs             ;try restart again

00D4    52 61 6D 20         erml:   db      'Ram Error'
00D8    45 72 72 6F
00DC    72
00DD    52 6F 6D 20         errm2:  db      'Rom Error'
00E1    45 72 72 6F
00E5    72
0009                        errml   equ     ($-erml)/2

                            ;;      initialize the interrupt vector table
                            ;
00E6    02                  intab:  defb    2
00E7    FF1A                        defw    sysvec+2
00E9    F140                        defw    keysrv          ;parallel keyboard interrupt vector

00EB    06                          defb    6
00EC    FF12                        defw    ctcvec+2
00EE    F1FD                        defw    milli           ;one millisecond interrupt timer
00F0    0000                        defw    0
00F2    F192                        defw    timer           ;one second timer interrupt vector
                            ;
                            ;       init keyboard fifo
                            ;
00F4    03                          defb    3
00F5    FF30                        defw    fifcnt
00F7    00                          defb    0               ;fifo count
00F8    00                          defb    0               ;fifo in
00F9    00                          defb    0               ;fifo out
                            ;
                            ;       initialize the crt display
                            ;
00FA    08                          defb    8
00FB    FFAC                        defw    cursor
00FD    3000                        defw    crtmem          ;base address is 3000h
00FF    02                          defb    02              ;use non-blinking box cursor
```

```
0100   3000                    defw   crtmem          ;direct crt memory output address
0102   17                      defb   23              ;initial scroll base
0103   00                      defb   0               ;initial leadin
0104   00                      defb   0               ;initial attribute
                       ;
                       ;       Initialize configurable parameter addresses
                       ;
0105   0C                      defb   2*numcon
0106   FFBF                    defw   contbl          ;configure table address
0108   F10C            cfinit: defw   siomsk          ;printer output ready mask
010A   F10E                    defw   sioval          ;printer output ready value
010C   F115                    defw   xonenb          ;Xon / Xoff enable/disable (NOP or RET)
010E   FF54                    defw   steprt          ;step rate for wd1797
0110   FFCB                    defw   spare1
0112   FFCC                    defw   spare2
0006                   numcon  equ    ($-cfinit)/2

0114   04                      defb   2*2
0115   FF3C                    defw   availb
0117   F7FD                    defw   iobloc+iobdvs
0119   FC80                    defw   ram-280h

011B   FF                      defb   -1              ;end of variable init table

                       ;;      I/O port initialization.
                       ;
                       ;       initialize system pio for use as bank-switch,
                       ;       configuration select and parallel keyboard input
                       ;

011C   01 1D                   defb   1,sysctl
011E   4F                      defb   01001111b       ;select input mode

011F   01 1C                   defb   1,syspio
0121   80                      defb   10000000b       ;enable ROM

0122   03 1D                   defb   3,sysctl
0124   CF                      defb   11001111b       ;put system pio in bit mode
0125   3F                      defb   00111111b       ;make bits 5, 4, 3, 2, 1,  and 0 be inputs
0126   07                      defb   00000111b       ;disable interrupts

0127   03 1F                   defb   3,kbdctl
0129   4F                      defb   01001111b       ;put keyboard port in input mode
012A   1A                      defb   sysvec+2        ;load keyboard interrupt vector
012B   83                      defb   10000011b       ;enable interrupts
                       ;
                       ;       Initialize Counter Timer Circuit.
                       ;
012C   02 18                   defb   2,ctc0
012E   03                      defb   00000011b       ;reset timer
012F   10                      defb   low ctcvec      ;base interrupt vector for ctc

0130   02 19                   defb   2,ctc1
0132   07                      defb   00000111b       ;start timer, but no interrupts
0133   FA                      defb   250             ;ctc1 period = 1 msec

```

```
0134    02 1A                  defb    2,ctc2
0136    27                     defb    00100111b       ;put ctc2 in timer / 256 mode (64 usec/count)
0137    7D                     defb    125             ;ctc2 period = 8 msec

0138    02 1B                  defb    2,ctc3
013A    C7                     defb    11000111b       ;put ctc3 in counter mode with interrupt
013B    7D                     defb    125             ;ctc3 period = 125*8 msec = 1 second
                       ;
                       ;       initialize sio channel b for asynchronous serial
                       ;       interface to printer or terminal
                       ;
013C    0A 07                  defb    10,siocpb
013E    01                     defb    1               ;select register #1
013F    00                     defb    00000000b       ;disable interrupts
0140    02                     defb    2               ;select register #2
0141    00                     defb    low siovec      ;base sio interrupt vector
0142    03                     defb    3               ;select register #3
0143    41                     defb    01000001b       ;7 bits/rx characters
0144    04                     defb    4               ;select register #4
0145    47                     defb    01000111b       ;16x clock, 1 stop bit, even parity enabled
0146    05                     defb    5               ;select register #5
0147    AA                     defb    10101010b       ;DTR, 7 bits/tx character, Tx enb, RTS

0148    01 0C                  defb    1,baudb
014A    07                     defb    0111b           ;default clock is 1200 bps

                       ;
                       ;       initialize communications port for async modem interface
                       ;
014B    08 06                  defb    8,siocpa
014D    01                     defb    1               ;select register #1
014E    00                     defb    00000000b       ;disable interrupts
014F    03                     defb    3               ;select register #3
0150    41                     defb    01000001b       ;7 bits/rx characters
0151    04                     defb    4               ;select register #4
0152    47                     defb    01000111b       ;16x clock, 1 stop bit, even parity enabled
0153    05                     defb    5               ;select register #5
0154    AA                     defb    10101010b       ;DTR, 7 bits/tx character, Tx enb, RTS

0155    01 00                  defb    1,bauda
0157    05                     defb    0101b           ;default clock is 300 bps

0158    01 68                  defb    1,async         ;set internal Rx+Tx clocks
015A    00                     defb    0
                       ;
                       ;       initialize PIO for Centronics style printer
                       ;
015B    03 09                  defb    3,gpioca
015D    CF                     defb    11001111b       ;mode 3
015E    00                     defb    00000000b       ;all output
015F    07                     defb    00000111b       ;no interrupts

0160    03 0B                  defb    3,gpiocb
0162    CF                     defb    11001111b       ;mode 3
0163    F0                     defb    11110000b       ;upper nibble in, lower out
0164    07                     defb    00000111b       ;no interrupts
```

```

      0165    01 0A                      defb    1,gpiodb
      0167    05                         defb    (1 shl p.strb) or (1 shl p.auto)

      0168    FF                         defb    -1              ;end of i/o init table

                                         subttl  Resident Monitor Entry Points
                                         page
```

```

                                  ;;      Resident monitor entry points.
                                  ;
                                  ;       This Entry Point Vector provides the only reliable access
                                  ;       to services provided by the Resident Monitor.  Any access
                                  ;       to code in the Monitor or its Ram page past the keyboard
                                  ;       variables is not allowed.  Future releases of the Resident
                                  ;       Monitor will always provide compatability with these entry
                                  ;       vectors.
                                  ;
                                  ;       This restriction also applies to the Resident Monitor Ram
                                  ;       Page at the top of memory.  Access to Ram Variables must
                                  ;       be obtained through the appropriate entry vector.
                                  ;
                                          above
        0266!                +            d&seg
        F000    C3 F07C           cold:   jp      restart         ;monitor restart
        F003    C3 FA62           warm:   jp      prompt          ;monitor entry point
        F006    C3 F0CD           const:  jp      kbdst           ;console status to A
        F009    C3 F0D8           conin:  jp      kbdin           ;console input to A
        F00C    C3 F2F1           conout: jp      crtout          ;console output from A
        F00F    C3 F2FE                   jp      fastcrt         ;fast crt output from C
        F012    C3 F0E5                   jp      siost           ;sio channel b status to A
        F015    C3 F0F0                   jp      sioin           ;sio channel b input to A
        F018    C3 F0F8                   jp      sioout          ;sio channel b output from A
        F01B    C3 FA17                   jp      select          ;select drive in C
        F01E    C3 FA3C                   jp      home            ;home r/w head
        F021    C3 FA3E                   jp      seek            ;seek to track in C
        F024    C3 FA48                   jp      read            ;read sector  C -> buffer @ HL
        F027    C3 FA44                   jp      write           ;write sector C <- buffer @ HL
        F02A    C3 F344                   jp      xqdvr           ;execute physical driver request @ HL
        F02D    C3 F284                   jp      setcur          ;set direct crt cursor from HL
        F030    C3 F288                   jp      outcur          ;direct crt display
        F033    C3 F2A3                   jp      crtldir         ;crt memory block move ala' LDIR
        F036    C3 F097                   jp      getsel          ;return address of disk mapping table to HL
        F039    C3 F086           dayti:  jp      daytim          ;return address of Time-of-Day
        F03C    C3 F08B                   jp      config          ;return configuration status
        F03F    C3 F105                   jp      siordy          ;sio channel b output ready status
        F042    C3 F0A4                   jp      setcon          ;set configuration
        F045    C3 F0BF                   jp      ssp             ;start screen print
        F048    C3 F13F           usrsec: jp      nulint          ;user accessible 1 second interrupt
        F04B    C3 F7A3                   jp      iocons          ;console status through iobyte
        F04E    C3 F7AF                   jp      ioconi          ;console input  through iobyte
        F051    C3 F796                   jp      iocono          ;console output through iobyte
        F054    C3 F7BB                   jp      iolist          ;printer output through iobyte
        F057    C3 F7CC                   jp      iolsts          ;printer status through iobyte
        F05A    C3 F770                   jp      comins          ;communications input ready status
        F05D    C3 F775                   jp      cominp          ;communications input data to A
        F060    C3 F77F                   jp      comout          ;communications output data from C
        F063    C3 F788                   jp      comots          ;communications output ready status
        F066    C3 F13F           idle:   jp      nulint          ;idle while i/o is pending
        F069    C3 F0D2           softv:  jp      soft            ;record soft error
        F06C                              defs    16,-1           ;space for option rom linkage

                                          subttl  Monitor Function Processors
                                          page
```

```

                                ;;      Monitor Restart.
                                ;
F07C    F3                      restart:di                      ;lock system
F07D    DB 1C                           in      a,(syspio)
F07F    F6 80                           or      1 shl 7         ;enable banked rom
F081    D3 1C                           out     (syspio),a
F083    C3 0000                         jp      prs             ;reload monitor from rom or ram

                                ;;      Daytim - Return Address of Time-of-Day.
                                ;
F086    11 FF56                 daytim: ld      de,day          ;point to day of month
F089    18 0F                           jr      retval

                                ;;      Config - Return Configuration Status Byte.
                                ;
F08B    3A F0E3                 config: ld      a,(mask)        ;turn keyboard mask into c.keym
F08E    E6 80                           and     080h
F090    F6 00                           or      0
F091                            confg   equ     $-1     ;*****=>;This word stored by Preset
F092    5F                              ld      e,a
F093    16 01                           ld      d,rev-400       ;return revision level
F095    18 03                           jr      retval

                                ;;      getsel - Get address of Select table.
                                ;
F097    11 F360                 getsel: ld      de,Seltab       ;set select table address

                                ;;      Retval - Return Value to Caller.
                                ;
F09A    24                      retval: inc     h               ;see if high level language call
F09B    25                              dec     h
F09C    28 03                           jr      z,retv1         ;if assembly level call
F09E    73                              ld      (hl),e          ;store answer in variable
F09F    23                              inc     hl
F0A0    72                              ld      (hl),d
F0A1    EB                      retv1:  ex      de,hl           ;leave result in HL as well
F0A2    FB                      eiret:  ei
F0A3    C9                              ret

                                ;;      setcon - set configuration.
                                ;
F0A4    7E                      setcon: ld      a,(hl)          ;get configuration table index
F0A5    CB BF                           res     7,a
F0A7    FE 06                           cp      numcon
F0A9    D0                              ret     nc              ;if index out of range
F0AA    5F                              ld      e,a
F0AB    7E                              ld      a,(hl)          ;get read/write flag
F0AC    23                              inc     hl
F0AD    46                              ld      b,(hl)          ;get configuration data
F0AE    16 00                           ld      d,0
F0B0    21 FFBF                         ld      hl,contbl       ;set address of configuration table addresses
F0B3    19                              add     hl,de
F0B4    19                              add     hl,de
F0B5    5E                              ld      e,(hl)          ;get configurable byte address
```

```
F0B6    23                      inc     hl
F0B7    56                      ld      d,(hl)
F0B8    EB                      ex      de,hl
F0B9    CB 7F                   bit     7,a             ;set direction
F0BB    7E                      ld      a,(hl)          ;get previous value
F0BC    C8                      ret     z               ;if asking current configuration
F0BD    70                      ld      (hl),b          ;store new configuration
F0BE    C9                      ret

                        ;;      ssp - start screen print.
                        ;
F0BF    3E 67           ssp:    ld      a,3+((24+1) shl 2)      ;start with cr/lf
F0C1    32 F20E                 ld      (spact),a
F0C4    AF                      xor     a
F0C5    32 F224                 ld      (spcnt),a
F0C8    3E 81                   ld      a,81h           ;start millisecond timer
F0CA    D3 19                   out     (ctc1),a
F0CC    C9                      ret

                                subttl  Console / Printer Drivers
                                page
```

```

                                          above                       ;run this code upstairs
        00CD"                  +          d&seg

                                  ;;      kbdst - keyboard status.
                                  ;
                                  ;       Returns A =  0 if no char
                                  ;               A = -1 if char available
                                  ;
        F0CD     3A FF30          kbdst:  ld      a,(fifcnt)          ;get input fifo bytecount
        F0D0     B7                       or      a
        F0D1     C8                       ret     z                   ;if keyboard queue is empty

                                  ;;      soft - record soft error.
                                  ;
        F0D2     F6 FF            soft:   or      -1                  ;set ready / error status
        F0D4     C9                       ret

                                  ;;      kbdin - Keyboard Input.
                                  ;
                                  ;       Returns A = character
                                  ;
        F0D5     CD F066          kbdin1: call    idle                ;idle cpu
        F0D8     CD F0CD          kbdin:  call    kbdst
        F0DB     28 F8                    jr      z,kbdin1            ;loop until keyboard input ready
        F0DD     E5                       push    hl
        F0DE     CD F130                  call    remove              ;get keyboard entry
        F0E1     E1                       pop     hl
        F0E2     E6 7F            kbmask: and     07fh
        F0E3                      mask    equ     $-1       ;*****=>;this byte modified by ESC 0/1
        F0E4     C9                       ret

                                  ;;      siost - sio channel b input ready status.
                                  ;
        F0E5     DB 07            siost:  in      a,(siocpb)          ;get sio status register
        F0E7     E6 01                    and     00000001b
        F0E9     C8                       ret     z                   ;if no data available
        F0EA     3E FF                    ld      a,-1
        F0EC     C9                       ret

                                  ;;      sioin - Sio channel b input character.
                                  ;
        F0ED     CD F066          sioin1: call    idle                ;idle cpu
        F0F0     CD F0E5          sioin:  call    siost               ;test console status
        F0F3     28 F8                    jr      z,sioin1            ;loop until data is
        F0F5     DB 05                    in      a,(siodpb)          ;ready at sio data port
        F0F7     C9                       ret

                                  ;;      sioout - Sio channel B output character.
                                  ;
        F0F8     F5               sioout: push    af
        F0F9     CD F105          siox1:  call    siordy
        F0FC     CC F066                  call    z,idle              ;idle cpu if transmitter not ready
        F0FF     28 F8                    jr      z,siox1
        F101     F1                       pop     af
```

```
F102    D3 05                   out     (siodpb),a      ;output data to sio
F104    C9                      ret

                        ;;      siordy - Sio channel B output ready ststus.
                        ;
F105    3E 10           siordy: ld      a,10h           ;reset status latch
F107    D3 07                   out     (siocpb),a
F109    DB 07                   in      a,(siocpb)
F10B    E6 04                   and     00000100b       ;test tbe status bit
F10C                    siomsk  equ     $-1     ;*****=>;modified at run time
F10D    EE 04                   xor     00000100b
F10E                    sioval  equ     $-1     ;*****=>;modified at run time
F10F    28 02                   jr      z,siord1        ;if hardware is ready
F111    AF                      xor     a
F112    C9                      ret
F113    F6 FF           siord1: or      -1              ;set ready status
F115    00              xonenb: nop             ;*****=>;put RET here to disable Xon/Xoff
F116    CD F0E5                 call    siost
F119    28 11                   jr      z,siord3        ;if input not available
F11B    CD F0F0                 call    sioin
F11E    E6 7F                   and     7fh
F120    D6 13                   sub     Xoff
F122    28 05                   jr      z,siord2        ;if printer said Stop
F124    D6 FE                   sub     Xon-Xoff
F126    20 04                   jr      nz,siord3       ;if not Resume
F128    2F                      cpl                     ;set printer ready
F129    32 F12D         siord2: ld      (xofflg),a
F12C    3E FF           siord3: ld      a,-1
F12D                    xofflg  equ     $-1     ;*****=>;set ^S pending flag
F12E    B7                      or      a
F12F    C9                      ret

                        ;;      Remove - remove key from fifo.
                        ;
F130    21 FF30         remove: ld      hl,fifcnt       ;decrement fifo count
F133    35                      dec     (hl)
F134    21 FF32                 ld      hl,fifout       ;point hl to fifo output offset
F137    34              index:  inc     (hl)            ;advance fifo pointer
F138    CB A6                   res     4,(hl)          ;modulo 16
F13A    3E 20                   ld      a,low fifo
F13C    86                      add     a,(hl)          ;index into fifo by offset
F13D    6F                      ld      l,a
F13E    7E                      ld      a,(hl)          ;fetch character in fifo
F13F    C9              nulint: ret

                                subttl  Interrupt Service Routines
                                page
```

```

                                       ;;      isr - interrupt service routines.
                                       ;
                                       service macro
                                               ld      (savstk),sp     ;;;save user stack pointer and
                                               ld      sp,intstk       ;;;switch to local stack
                                               push    hl              ;;;save machine state
                                               push    af
                                               endm

                                       ;;      keysrv - parallel keyboard interrupt service.
                                       ;
        F140                           keysrv: service                 ;save state
        F140    ED 73 F1EC      +              ld      (savstk),sp
        F144    31 FF50         +              ld      sp,intstk
        F147    E5              +              push    hl
        F148    F5              +              push    af
        F149    C5                             push    bc
        F14A    DB 1E                          in      a,(kbddat)      ;read keyboard input port
        F14C    2F                             cpl
        F14D    FE 9E                          cp      Scrprt
        F14F    20 16                          jr      nz,key2         ;if not screen print key
        F151    3A F20E                        ld      a,(spact)
        F154    B7                             or      a
        F155    28 0B                          jr      z,key1          ;if screen not printing now
        F157    3E 07                          ld      a,3+(1 shl 2)   ;set state to cr/lf/stop
        F159    32 F20E                        ld      (spact),a       ;set screen print state
        F15C    AF                             xor     a
        F15D    32 F224                        ld      (spcnt),a       ;restart character counter
        F160    18 2D                          jr      key5
        F162    CD F0BF                key1:   call    ssp             ;start screen print
        F165    18 28                          jr      key5
        F167    21 FF30                key2:   ld      hl,fifcnt       ;bump input fifo character count
        F16A    FE 9B                          cp      Abort           ;check user Abort Key
        F16C    28 11                          jr      z,key3          ;warm start system
        F16E    4F                             ld      c,a
        F16F    7E                             ld      a,(hl)
        F170    3C                             inc     a
        F171    FE 10                          cp      16
        F173    30 1A                          jr      nc,key5         ;exit now if fifo is full
        F175    77                             ld      (hl),a
        F176    21 FF31                        ld      hl,fifin        ;point hl to fifo input offset
        F179    CD F137                        call    index
        F17C    71                             ld      (hl),c          ;store character in fifo @ hl
        F17D    18 10                          jr      key5
        F17F    CD F1EF                key3:   call    retins          ;release Pio interrupt controller
        F182    06 03                          ld      b,3
        F184    36 00                  key4:   ld      (hl),0          ;clear fifo count
        F186    2C                             inc     l               ;and fifo in/out pointers
        F187    10 FB                          djnz    key4
        F189    CD F293                        call    crtoff          ;turn crt memory off
        F18C    CD F003                        call    warm            ;and warm start system
        F18F    C1                     key5:   pop     bc
        F190    18 57                          jr      rfi             ;return from interrupt

```

```
                                ;;      timer - Once per second interrupt.
                                ;
F192                            timer:  service
F192    ED 73 F1EC      +               ld      (savstk),sp
F196    31 FF50         +               ld      sp,intstk
F199    E5              +               push    hl
F19A    F5              +               push    af
F19B    2A FF52                         ld      hl,(tikcnt)     ;advance binary seconds counter
F19E    23                              inc     hl
F19F    22 FF52                         ld      (tikcnt),hl
F1A2    CD F048                         call    usrsec          ;invoke user's interrupt routine
F1A5    21 FF55                         ld      hl,timout       ;decrement disk turn-off timer
F1A8    35                              dec     (hl)
F1A9    20 06                           jr      nz,timer1       ;exit if not timed out yet
F1AB    DB 1C                           in      a,(syspio)
F1AD    E6 F8                           and     11111000b       ;disable all drive selects which
F1AF    D3 1C                           out     (syspio),a      ;turns off spindle motors
F1B1    C5                      timer1: push    bc
F1B2    06 02                           ld      b,2
F1B4    3E 3B                           ld      a,59
F1B6    21 FF5B                         ld      hl,secs         ;point at wall clock
F1B9    34                      timer2: inc     (hl)            ;increment seconds
F1BA    BE                              cp      (hl)
F1BB    30 2B                           jr      nc,timer3       ;if not one minute or hour
F1BD    36 00                           ld      (hl),0
F1BF    2B                              dec     hl
F1C0    10 F7                           djnz    timer2
F1C2    3E 17                           ld      a,23
F1C4    34                              inc     (hl)            ;increment hours
F1C5    BE                              cp      (hl)
F1C6    30 20                           jr      nc,timer3       ;if not one day
F1C8    36 00                           ld      (hl),0
F1CA    D5                              push    de
F1CB    2B                              dec     hl
F1CC    2B                              dec     hl
F1CD    4E                              ld      c,(hl)          ;get month
F1CE    2B                              dec     hl              ;point to day
F1CF    11 F1F0                         ld      de,dpm-1        ;point to day/month table
F1D2    EB                              ex      de,hl
F1D3    09                              add     hl,bc
F1D4    7E                              ld      a,(hl)          ;get number of days
F1D5    EB                              ex      de,hl
F1D6    D1                              pop     de
F1D7    34                              inc     (hl)            ;increment day
F1D8    BE                              cp      (hl)
F1D9    30 0D                           jr      nc,timer3       ;if not end of month
F1DB    36 01                           ld      (hl),1          ;reset day in month
F1DD    23                              inc     hl
F1DE    34                              inc     (hl)            ;increment month
F1DF    3E 0B                           ld      a,11
F1E1    BE                              cp      (hl)
F1E2    30 04                           jr      nc,timer3       ;if not new years eve
F1E4    36 01                           ld      (hl),1          ;wrap december to january
F1E6    23                              inc     hl
F1E7    34                              inc     (hl)            ;signal Guy Lombardo
F1E8    C1                      timer3: pop     bc
```

```
                                 ;;      rfi - return From Interrupt.
                                 ;
F1E9    F1                       rfi:    pop     af
F1EA    E1                               pop     hl
F1EB    31 0000                          ld      sp,0             ;restore stack
F1EC                             savstk  equ     $-2     ;*****=>;this word modified at runtime
F1EE    FB                               ei                       ;re-enable interrupts and return
F1EF    ED 4D                    retins: reti

                                 ;;      Table of days per month.
                                 ;
F1F1    1F                       dpm:    db      31               ;january
F1F2    1C                               db      28               ;febuary
F1F3    1F                               db      31               ;march
F1F4    1E                               db      30               ;april
F1F5    1F                               db      31               ;may
F1F6    1E                               db      30               ;june
F1F7    1F                               db      31               ;july
F1F8    1F                               db      31               ;august
F1F9    1E                               db      30               ;september
F1FA    1F                               db      31               ;october
F1FB    1E                               db      30               ;november
F1FC    1F                               db      31               ;december

                                 ;;      milli - Millisecond timer interrupt service.
                                 ;
F1FD                             milli:  service
F1FD    ED 73 F1EC       +               ld      (savstk),sp
F201    31 FF50          +               ld      sp,intstk
F204    E5               +               push    hl
F205    F5               +               push    af
F206    2A FF50                          ld      hl,(Milsec)
F209    23                               inc     hl               ;increment millisecond counter
F20A    22 FF50                          ld      (Milsec),hl
F20D    3E 00                            ld      a,0              ;set screen print flag
F20E                             spact   equ     $-1     ;*****=>;this byte modified at runtime
F20F    67                               ld      h,a
F210    E6 03                            and     3
F212    28 6D                            jr      z,mil16          ;if not printing screen
F214    6F                               ld      l,a
F215    CD F105                          call    siordy           ;get printer status
F218    28 67                            jr      z,mil16          ;if printer not ready
F21A    2D                               dec     l
F21B    20 48                            jr      nz,mil12         ;if not character print state
F21D    DB 1C                            in      a,(syspio)       ;get pio state
F21F    F5                               push    af
F220    CD F29C                          call    crton
F223    3E 00                            ld      a,0              ;get character count
F224                             spcnt   equ     $-1     ;*****=>;byte modified at runtime
F225    3D                               dec     a
F226    FA F236                          jp      m,mil0           ;if end of line
F229    32 F224                          ld      (spcnt),a
F22C    21 0000                          ld      hl,0             ;set next character address
F22D                             spaddr  equ     $-2     ;*****=>;word modified at runtime
F22F    7E                               ld      a,(hl)
```

```
F230    D3 05                   out     (siodpb),a       ;fire hammer
F232    2C                      inc     l                ;advance screen cursor
F233    AF                      xor     a                ;do not advance state
F234    18 22                   jr      mil11            ;if not end of line
F236    3E 61           mil0:   ld      a,1+(24 shl 2)   ;set address of next print line
F238    94                      sub     h
F239    1F                      rra
F23A    CB 2F                   sra     a
F23C    CD F31E                 call    cca              ;compute cursor address
F23F    E5                      push    hl               ;save next line address
F240    C5                      push    bc
F241    06 50                   ld      b,80             ;delete  trailing blanks
F243    7D                      ld      a,l
F244    80                      add     a,b
F245    6F                      ld      l,a
F246    2D              mil01:  dec     l
F247    7E                      ld      a,(hl)           ;get next character
F248    E6 7F                   and     7fh
F24A    FE 20                   cp      ' '
F24C    20 02                   jr      nz,mil02         ;if not trailing blank
F24E    10 F6                   djnz    mil01
F250    78              mil02:  ld      a,b
F251    32 F224                 ld      (spcnt),a        ;set number of characters to print
F254    C1                      pop     bc
F255    E1                      pop     hl
F256    3E 03                   ld      a,3              ;set CR next state
F258    22 F22D         mil11:  ld      (spaddr),hl      ;set next display address
F25B    21 F20E                 ld      hl,spact         ;set state variable
F25E    B6                      or      (hl)             ;advance state
F25F    77                      ld      (hl),a
F260    F1                      pop     af               ;get pio back
F261    D3 1C                   out     (syspio),a
F263    18 1C                   jr      mil16
F265    2D              mil12:  dec     l                ;check next state
F266    21 F20E                 ld      hl,spact         ;set state address
F269    20 11                   jr      nz,mil14         ;if not lf state
F26B    7E                      ld      a,(hl)
F26C    D6 04                   sub     1 shl 2          ;advance line counter
F26E    77                      ld      (hl),a
F26F    FE FE                   cp      2-(1 shl 2)
F271    20 05                   jr      nz,mil13
F273    3E 01                   ld      a,1              ;disable ctc interrupt
F275    D3 19                   out     (ctc1),a
F277    77                      ld      (hl),a
F278    3E 0A           mil13:  ld      a,lf             ;set line feed
F27A    18 02                   jr      mil15
F27C    3E 0D           mil14:  ld      a,cr             ;set carriage return
F27E    D3 05           mil15:  out     (siodpb),a       ;move paper or carriage
F280    35                      dec     (hl)
F281    C3 F1E9         mil16:  jp      rfi              ;return from interrupt

                                subttl  Crt Driver
                                page
```

```

                                ;;      setcur - set direct display cursor position.
                                ;
F284    22 FFAF                 setcur: ld      (dircur),hl     ;set up cursor address
F287    C9                              ret

                                ;;      outcur - store character directly to crt memory.
                                ;
F288    CD F29C                 outcur: call    crton           ;turn on crt bank
F28B    2A FFAF                         ld      hl,(dircur)     ;fetch direct cursor
F28E    71                              ld      (hl),c          ;store character
F28F    23                              inc     hl
F290    22 FFAF                         ld      (dircur),hl

                                ;;      crtoff - turn crt ram off.
                                ;
F293    F3                      crtoff: di                      ;lock pio access
F294    DB 1C                           in      a,(syspio)
F296    CB BF                   crtofl: res     7,a             ;reset crt bank enable
F298    FB                              ei                      ;unlock pio access
F299    D3 1C                   crtonl: out     (syspio),a
F29B    C9                              ret

                                ;;      crton - turn crt ram on.
                                ;
F29C    F3                      crton:  di                      ;lock time-out interrupt
F29D    DB 1C                           in      a,(syspio)      ;get pio status
F29F    CB FF                           set     7,a             ;enable bank
F2A1    18 F6                           jr      crtonl

                                ;;      block move from/to crt memory.
                                ;
                                ;       Entry:  HL = Source address
                                ;               DE = Destination address
                                ;               BC = Number of bytes to move
                                ;               A  = 0 - Move crt ram to crt ram
                                ;               A  < 0 - Move sys ram to crt ram
                                ;               A  > 0 - Move crt ram to sys ram
                                ;
F2A3    ED 73 F31B              crtldir:ld      (usrstk),sp     ;do not use callers stack
F2A7    31 FFE0                         ld      sp,crtstk       ;since it may disappear
F2AA    A7                              and     a               ;set entry conditions
F2AB    CD F29C                         call    crton
F2AE    28 37                           jr      z,crtmv         ;block move within crt ram
F2B0    F2 F2B5                         jp      p,ldir2         ;if move from crt ram to system ram
F2B3    EE 80                   ldirl:  xor     80h
F2B5    D3 1C                   ldir2:  out     (syspio),a      ;enable source bank
F2B7    E5                              push    hl              ;save move source address
F2B8    21 FFB0                         ld      hl,-80          ;count down one transfer buffer
F2BB    ED 4A                           adc     hl,bc
F2BD    E3                              ex      (sp),hl         ;save overflow, retrieve source address
F2BE    FA F2C4                         jp      m,ldir3         ;if less than one buffer
F2C1    01 0050                         ld      bc,80           ;transfer one buffer
F2C4    C5                      ldir3:  push    bc              ;save byte count
F2C5    D5                              push    de              ;save destination address
```

```
F2C6    11 FF5C                  ld      de,linbuf       ;set upper buffer
F2C9    ED B0                    ldir                    ;move data to upper ram
F2CB    D1                       pop     de
F2CC    C1                       pop     bc
F2CD    DB 1C                    in      a,(syspio)      ;enable destination bank
F2CF    EE 80                    xor     80h
F2D1    D3 1C                    out     (syspio),a
F2D3    E5                       push    hl              ;save source address
F2D4    21 FF5C                  ld      hl,linbuf       ;set upper  buffer
F2D7    ED B0                    ldir                    ;move data from buffer to destination
F2D9    E1                       pop     hl
F2DA    C1                       pop     bc              ;retrieve bytes left to transfer
F2DB    78                       ld      a,b
F2DC    A7                       and     a
F2DD    FA F2E9                  jp      m,crtmvo        ;no more move, turn crt ram off and return
F2E0    B1                       or      c
F2E1    28 06                    jr      z,crtmvo        ;if no more
F2E3    DB 1C                    in      a,(syspio)
F2E5    18 CC                    jr      ldir1           ;continue transfer one buffer at a crack

F2E7    ED B0           crtmv:   ldir
F2E9    CD F293         crtmvo:  call    crtoff          ;turn crt ram off
F2EC    ED 7B F31B               ld      sp,(usrstk)
F2F0    C9                       ret

                                 subttl  Resident Crt Driver.
                                 page
```

```
                                 ;;      crtout - Crt Output Driver.
                                 ;
                                 ;       Entry:  Character in register A
                                 ;               16 bytes of stack space available
                                 ;       Exit:   Char displayed, all registers saved
                                 ;
F2F1    E5                       crtout: push    hl              ;maintain users registers on his stack
F2F2    D5                               push    de
F2F3    C5                               push    bc
F2F4    F5                               push    af
F2F5    4F                               ld      c,a             ;set character to process
F2F6    CD F2FE                          call    fastcrt         ;process character quickly
F2F9    F1                               pop     af              ;restore callers registers
F2FA    C1                               pop     bc
F2FB    D1                               pop     de
F2FC    E1                               pop     hl
F2FD    C9                               ret
                                 ;;      fastcrt - fast crt driver.
                                 ;
                                 ;       Entry:  Character in C
                                 ;       Exit:   The only register preserved is SP
                                 ;               Peeking in register A reveals valuable characters.
                                 ;
F2FE    ED 73 F31B               fastcrt:ld      (usrstk),sp     ;do not use callers stack
F302    31 FFE0                          ld      sp,crtstk       ;since it may disappear
F305    DD E5                            push    ix
F307    DB 1C                            in      a,(syspio)      ;read system pio
F309    B7                               or      a               ;set bank enable status
F30A    F5                               push    af              ;save status for exit code
F30B    CD F29C                          call    crton           ;turn on crt memory
F30E    FB                               ei                      ;enable interrupts
F30F    CD 0169                          call    crtdvr          ;execute crt driver rom
F312    F1                               pop     af              ;get previous bank enable status
F313    F4 F293                          call    p,crtoff        ;disable bank now if it was disabled on entry
F316    DD E1                            pop     ix
F318    3E 00                            ld      a,0             ;sneak balcones golden characters to FAST users
F319                             gold    equ     $-1
F31A    31 F31B                          ld      sp,usrstk       ;restore callers stack
F31B                             usrstk  equ     $-2     ;*****=>;this operand word is modified at runtime
F31D    C9                               ret
                                 ;;      cca - compute cursor address.
                                 ;
                                 ;       Entry:  A = Row
                                 ;
F31E    67                       cca:    ld      h,a
F31F    3A FFB1                          ld      a,(base)
F322    84                               add     a,h
F323    3C                       cca1:   inc     a               ;entry with base absolute
F324    D6 18                    cca2:   sub     24              ;ditto
F326    30 FC                            jr      nc,cca2
F328    C6 78                            add     a,24+2*crtbas
F32A    67                               ld      h,a
```

```
F32B    2E 00                       ld      l,0
F32D    CB 2C                       sra     h
F32F    CB 1D                       rr      l
F331    C9                          ret

                            ;;      rstatt - Restore Previous Attribute.
                            ;
F332    01 0000             rstatt: ld      bc,0            ;execute previous attribute routine
F333                        lstatt  equ     $-2
F335    C5                          push    bc
F336    C9                          ret

F337    E5                  setprv: push    hl
F338    21 01CF                     ld      hl,setlow
F339                        prvatt  equ     $-2
F33B    22 F333                     ld      (lstatt),hl
F33E    ED 43 F339                  ld      (prvatt),bc
F342    E1                          pop     hl
F343    C9                          ret

                                    subttl  Rom-resident Crt Driver
                                    page
```

```

                                  below                     ;execute in banked rom
0000!                +            defs    comres

                          ;;      crtdvr - Crt Driver Proper.
                          ;
0169    2A FFAC           crtdvr: ld      hl,(cursor)       ;set cursor address
016C    3A FFB4                   ld      a,(chrsav)        ;retrieve character under cursor
016F    77                        ld      (hl),a            ;replace character under cursor
0170    32 F319                   ld      (gold),a          ;bury balcones gold
0173    3A FFB2                   ld      a,(leadin)        ;set leadin state
0176    B7                        or      a
0177    20 1D                     jr      nz,crtd2          ;if processing escape sequence
0179    3A F0E3                   ld      a,(mask)          ;get keyboard mask
017C    A1                        and     c
017D    4F                        ld      c,a
017E    FE 20                     cp      ' '
0180    38 14                     jr      c,crtd2           ;if control code
0182    3A FFB3           crtd1:  ld      a,(attrib)
0185    B1                        or      c
0186    77                        ld      (hl),a            ;store displayable character
0187    2C                        inc     l                 ;advance pointer to next column
0188    7D                        ld      a,l
0189    E6 7F                     and     01111111b         ;extract column# from hl
018B    FE 50                     cp      80
018D    38 0A                     jr      c,crtd3           ;if end of line not reached
018F    AD                        xor     l
0190    6F                        ld      l,a               ;return cursor to left side
0191    CD 02F7                   call    lfeed             ;execute line feed
0194    18 03                     jr      crtd3
0196    CD 023D           crtd2:  call    contrl            ;process control character
0199    22 FFAC           crtd3:  ld      (cursor),hl       ;save cursor pointer for next time
019C    7E                        ld      a,(hl)            ;get character at new cursor location
019D    32 FFB4                   ld      (chrsav),a        ;save for next time 'CRTOUT' is called
01A0    3A FFAE                   ld      a,(csrchr)        ;get cursor character
01A3    FE 20                     cp      ' '
01A5    C8                        ret     z                 ;if no cursor
01A6    4F                        ld      c,a
01A7    7E                        ld      a,(hl)
01A8    CB BF                     res     7,a
01AA    FE 20                     cp      ' '
01AC    79                        ld      a,c               ;set character used for cursor
01AD    28 03                     jr      z,crtd4           ;if character is a space
01AF    7E                        ld      a,(hl)            ;toggle attribute
01B0    EE 80                     xor     80h
01B2    77                crtd4:  ld      (hl),a            ;store cursor character
01B3    C9                        ret

                          ;;      multi - Process multiple character escape sequence.
                          ;
01B4    EB                multi:  ex      de,hl             ;unconditionally reset the lead-in
01B5    36 00                     ld      (hl),0            ;state to zero
01B7    EB                        ex      de,hl
01B8    3D                        dec     a
01B9    20 4E                     jr      nz,setxy1         ;if not initial state
```

```
01BB    79                      ld      a,c             ;get second character of sequence

                                if      (options and o.esct) ne 0
                        ;;      Escape table search
                        ;
01BC    E5                      push    hl
01BD    21 028A                 ld      hl,esctab
01C0    01 0011                 ld      bc,esctbl
01C3    ED B1           search: cpir
01C5    09                      add     hl,bc
01C6    09                      add     hl,bc
01C7    09                      add     hl,bc
01C8    4E                      ld      c,(hl)
01C9    23                      inc     hl
01CA    46                      ld      b,(hl)
01CB    E1                      pop     hl
01CC    C0                      ret     nz
01CD    C5                      push    bc
01CE    C9                      ret
                                endif                   ;options and o.esct

                        ;;      Set attribute modes.
                        ;
01CF    D3 36           setlow: out     (lowlite),a     ;set lo-light mode
01D1    C3 F337                 jp      setprv

01D4    D3 35           setbli: out     (chrom2),a      ;select rom 2
01D6    AF                      xor     a               ;select standard char set
01D7    18 08                   jr      mode1

01D9    D3 35           setinv: out     (chrom2),a      ;select rom 2
01DB    18 02                   jr      mode

01DD    D3 34           setgra: out     (chrom1),a      ;select rom 1
01DF    3E 40           mode:   ld      a,40h           ;select alternate char set
01E1    CD F337         mode1:  call    setprv          ;set up previous attribute
01E4    47                      ld      b,a
01E5    F3                      di                      ;lock system
01E6    DB 1C                   in      a,(syspio)      ;read system pio
01E8    CB B7                   res     6,a             ;clear rom select bit
01EA    B0                      or      b
01EB    FB                      ei                      ;unlock
01EC    D3 1C                   out     (syspio),a      ;set or reset display mode
01EE    C9                      ret

                        ;;      Enable/disable (D7) display of selected attribute mode
                        ;
01EF    3E 80           enatr:  ld      a,80h
01F1    06                      db      6 ;ld b,        ;skip xor
01F2    AF              disatr: xor     a
01F3    32 FFB3         dis1:   ld      (attrib),a
01F6    C9                      ret

                        ;;      setmsk - Select 7 or 8 bit data  from keyboard
                        ;
01F7    0F              setmsk: rrca                    ;get low order bit as upper bit mask
```

```
01F8    F6 7F              or      7fh
01FA    4F                 ld      c,a
01FB    11 F0E3            ld      de,mask
01FE    1A                 ld      a,(de)
01FF    32 F319            ld      (gold),a        ;stash balcones gold
0202    79                 ld      a,c
0203    12                 ld      (de),a          ;store keyboard mask
0204    C9                 ret

                   ;;      Process cursor position sequence.
                   ;
0205    3E 02      setxy:  ld      a,2
0207    12                 ld      (de),a          ;make leadin=2 next time
0208    C9                 ret

0209    3D         setxy1: dec     a
020A    20 10              jr      nz,m3tst        ;if not in state 2
020C    6F                 ld      l,a             ;clear low cursor pos
020D    3E 03              ld      a,3
020F    12                 ld      (de),a          ;set state 3 for next time
0210    79         setrow: ld      a,c
0211    E6 7F              and     07fh            ;strip parity bit
0213    D6 20              sub     ' '
0215    D8                 ret     c               ;if illegal character
0216    FE 18              cp      24
0218    D0                 ret     nc
0219    C3 F31E            jp      cca             ;compute cursor address

021C    3D         m3tst:  dec     a
021D    20 0C              jr      nz,m4tst        ;if not ready for column
021F    79         setcol: ld      a,c
0220    E6 7F              and     07fh            ;strip parity bit
0222    D6 20              sub     ' '             ; of esc,'=',row,col sequence
0224    D8                 ret     c
0225    FE 50              cp      80
0227    D0                 ret     nc
0228    B5                 or      l               ;merge in col# with l
0229    6F                 ld      l,a
022A    C9                 ret

022B    3D         m4tst:  dec     a
022C    20 04              jr      nz,m5tst        ;if not escape state 4
022E    D1                 pop     de              ;pitch address of crtd3
022F    C3 0182            jp      crtd1           ;display character in C

0232    3A FFAE    m5tst:  ld      a,(csrchr)
0235    32 F319            ld      (gold),a        ;stash balcones gold
0238    79                 ld      a,c
0239    32 FFAE            ld      (csrchr),a      ;store new cursor character
023C    C9                 ret

                   ;;      contrl - process control character.
                   ;
023D    11 FFB2    contrl: ld      de,leadin       ;point at leadin state
0240    D2 01B4            jp      nc,multi        ;if multi code sequence in progress
0243    FE 05              cp      'E'-64
```

```
0245    D8                      ret     c               ;control char is out of range
0246    E5                      push    hl              ;save cursor
0247    21 024A                 ld      hl,ctltab-('E'-64)*2
024A    06 00                   ld      b,0
024C    09                      add     hl,bc           ;index through control character table
024D    09                      add     hl,bc
024E    4E                      ld      c,(hl)
024F    23                      inc     hl
0250    46                      ld      b,(hl)          ;get address of control subroutine
0251    E1                      pop     hl
0252    C5                      push    bc
0253    C9                      ret                     ;execute control code driver

                                if      (options and o.esct) ne 0
0254    02C5            ctltab: defw    defcur          ;Ctrl-e is define new cursor character
0256    F332                    defw    rstatt          ;Ctrl-f is restore previous attribute mode
0258    032F                    defw    bell            ;Ctrl-g is the bell
025A    02CE                    defw    bakspc          ;Ctrl-h is cursor left
025C    031F                    defw    tab             ;Ctrl-i is tab
025E    02F7                    defw    lfeed           ;Ctrl-j is cursor down
0260    02DC                    defw    upcsr           ;Ctrl-k is cursor up
0262    02D4                    defw    forspc          ;Ctrl-l is cursor right
0264    02F2                    defw    return          ;Ctrl-m is carriage return
0266    02C0                    defw    nono            ;Ctrl-n is not acceptable
0268    02C0                    defw    nono            ;Ctrl-o is not acceptable
026A    02C0                    defw    nono            ;Ctrl-p is not acceptable
026C    0361                    defw    clreos          ;Ctrl-q is clear to end-of-screen
026E    02C0                    defw    nono            ;Ctrl-r is not acceptable
0270    02C0                    defw    nono            ;Ctrl-s is not acceptable
0272    02C0                    defw    nono            ;Ctrl-t is not acceptable
0274    02C0                    defw    nono            ;Ctrl-u is not acceptable
0276    02C0                    defw    nono            ;Ctrl-v is not acceptable
0278    02C0                    defw    nono            ;Ctrl-w is not acceptable
027A    0344                    defw    clreol          ;Ctrl-x is clear to end-of-line
027C    02C0                    defw    nono            ;Ctrl-y is not acceptable
027E    0357                    defw    clrscn          ;Ctrl-z is clear screen
0280    02BD                    defw    escape          ;Ctrl-[ is escape
0282    02C0                    defw    nono            ;Ctrl-\ is not acceptable
0284    02C0                    defw    nono            ;Ctrl-] is not acceptable
0286    02C9                    defw    homeup          ;Ctrl-^ is home up
0288    02C1                    defw    stuff           ;Ctrl-_ is display control chars

0036                    ctlsiz  equ     $-ctltab

                        ;;      Escape sequence table.
                        ;
                        ;       Maintains functional compatibility with terminals supporting
                        ;       ADM-3a style supersets.
                        ;
028A    28              esctab: db      '('             ;disable attribute
028B    29                      db      ')'             ;enable atribute
028C    2A                      db      '*'             ;clear screen
028D    30                      db      '0'             ;strip keyboard upper bit
028E    31                      db      '1'             ;pass keyboard upper bit
028F    34                      db      '4'             ;char font and blinking
0290    35                      db      '5'             ;char font and graphics
```

```
0291    36                     db      '6'               ;char font and blinking
0292    37                     db      '7'               ;char font and inverse video
0293    38                     db      '8'               ;char font and lo-light
0294    3D                     db      '='               ;position cursor
0295    45                     db      'E'               ;line insert
0296    51                     db      'Q'               ;character insert
0297    52                     db      'R'               ;line delete
0298    57                     db      'W'               ;character delete
0299    54                     db      'T'               ;clear to end of line
029A    59                     db      'Y'               ;clear to end of screen
0011                   esctbl  equ     $-esctab

029B    0361           escadr: defw    clreos
029D    0344                   defw    clreol
029F    03F5                   defw    chrdel
02A1    037C                   defw    lindel
02A3    03DC                   defw    chrins
02A5    03A4                   defw    linins
02A7    0205                   defw    setxy
02A9    01CF                   defw    setlow
02AB    01D9                   defw    setinv
02AD    01D4                   defw    setbli
02AF    01DD                   defw    setgra
02B1    01D4                   defw    setbli
02B3    01F7                   defw    setmsk
02B5    01F7                   defw    setmsk
02B7    0357                   defw    clrscn
02B9    01EF                   defw    enatr
02BB    01F2                   defw    disatr
                               if1
                               if      ($-escadr)/2 ne esctbl
                               .printx Escape table mismatch
                               endif
                               endif
                               endif                     ;o.esct and options

                       ;;      escape - Initialize escape sequence.
                       ;
02BD    3E 01          escape: ld      a,1
02BF    12                     ld      (de),a            ;set sequence state
02C0    C9             nono:   ret                       ;for escape processing

                       ;;      stuff - Enable next char to be stored directly.
                       ;
02C1    3E 04          stuff:  ld      a,4
02C3    12                     ld      (de),a            ;set sequence state
02C4    C9                     ret                       ;for control char output mode

                       ;;      defcur - Enable next chara to be new cursor.
                       ;
02C5    3E 05          defcur: ld      a,5
02C7    12                     ld      (de),a
02C8    C9                     ret

                       ;;      homeup - Move cursor to upper left.
                       ;
```

```
02C9   0E 20                 homeup: ld    c,' '          ;use cursor addressing routine
02CB   C3 0210                       jp    setrow         ;to do homeup almost for free
                             ;;      bakspc - Move cursor left.
                             ;
02CE   7D                    bakspc: ld    a,l            ;check for left margin
02CF   E6 7F                         and   01111111b
02D1   C8                            ret   z              ;abort if in leftmost column
02D2   2D                            dec   l              ;back up cursor pointer
02D3   C9                            ret
                             ;;      forspc - Move cursor right.
                             ;
02D4   7D                    forspc: ld    a,l            ;check for rightmost colunm
02D5   E6 7F                         and   01111111b
02D7   FE 4F                         cp    79
02D9   D0                            ret   nc             ;do nothing if already there
02DA   2C                            inc   l
02DB   C9                            ret                  ;else advance the cursor pointer
                             ;;      upscr - Move cursor up.
                             ;
02DC   11 FF80               upcsr:  ld    de,-128        ;subtract 1 from row# component
02DF   19                            add   hl,de          ; of cursor pointer in hl
02E0   7C                            ld    a,h
02E1   FE 30                         cp    crtbas         ;check for underflow of pointer
02E3   D0                            ret   nc
02E4   26 3B                         ld    h,crttop-1     ;wrap cursor around modulo 3k
02E6   C9                            ret
                             ;;      dncsr - Move cursor down.
                             ;
02E7   11 0080               dncsr:  ld    de,128         ;add 1 to row# component
02EA   19                            add   hl,de          ; of cursor pointer in hl
02EB   7C                            ld    a,h
02EC   FE 3C                         cp    crttop         ;check for overflow of pointer
02EE   D8                            ret   c
02EF   26 30                         ld    h,crtbas       ;reset pointer modulo 128*24
02F1   C9                            ret
                             ;;      return - Move cursor to left side.
                             ;
02F2   7D                    return: ld    a,l            ;clear column
02F3   E6 80                         and   10000000b
02F5   6F                            ld    l,a            ;move cursor pointer back
02F6   C9                            ret                  ; to start of line
                             ;;      lfeed - Move cursor down with scroll.
                             ;
02F7   7D                    lfeed:  ld    a,l
02F8   17                            rla
02F9   7C                            ld    a,h
02FA   17                            rla                  ;extract row# component of hl
02FB   E6 1F                         and   00011111b
02FD   4F                            ld    c,a            ;copy row# into c for scroll test
02FE   CD 02E7                       call  dncsr          ;move cursor to next row down
```

```
0301    3A FFB1              ld      a,(base)        ;test if cursor was on bottom row
0304    91                   sub     c               ;of screen before moving down
0305    32 F319              ld      (gold),a        ;send scroll flag back to fast users
0308    C0                   ret     nz              ;exit if not at bottom
0309    E5                   push    hl              ;prepare scroll screen up
030A    3A F31C              ld      a,(usrstk+1)
030D    3C                   inc     a
030E    C4 040F              call    nz,bbg          ;bury balcones gold, unless monitor running
0311    CD 0341              call    clrlin          ;fill top line with spaces
0314    29                   add     hl,hl
0315    7C                   ld      a,h             ;get row# component of hl into a
0316    E6 1F                and     00011111b
0318    32 FFB1              ld      (base),a        ;store new base line#
031B    D3 14                out     (scroll),a      ;scroll top line to bottom
031D    E1                   pop     hl              ;restore cursor
031E    C9                   ret
                     ;;      tab - Move cursor to next tab stop.
                     ;
031F    11 0008      tab:    ld      de,8            ;tabs are every 8 columns
0322    7D                   ld      a,l             ;get column component of
0323    E6 78                and     01111000b       ;previous tab position
0325    83                   add     a,e
0326    FE 50                cp      80
0328    D0                   ret     nc              ;if next tab column past end of line
0329    7D                   ld      a,l
032A    E6 F8                and     11111000b       ;advance cursor to next tab stop
032C    6F                   ld      l,a
032D    19                   add     hl,de
032E    C9                   ret
                     ;;      bell - Move speaker back and forth.
                     ;
032F    3E 35        bell:   ld      a,bltim         ;Bell time constant
0331    D3 29        belll:  out     (bellon),a      ;push speaker out
0333    06 61                ld      b,blonc         ;set Bell on time constant
0335    10 FE                djnz    $               ;pause B*2 micro seconds
0337    D3 28                out     (bellof),a      ;yank speaker in
0339    06 61                ld      b,blofc         ;set Bell off time constant
033B    10 FE                djnz    $
033D    3D                   dec     a
033E    20 F1                jr      nz,belll        ;if more noise to make
0340    C9                   ret
                     ;;      clrlin - Clear line.
                     ;
0341    CD 02F2      clrlin: call    return          ;return cursor and fall through clreol
                     ;;      clreol - Clear to end of line.
                     ;
0344    7D           clreol: ld      a,l
0345    E6 7F                and     01111111b       ;get column component of cursor posistion
0347    FE 50                cp      80
0349    D0                   ret     nc              ;if someone busted curpos
034A    ED 44                neg
034C    C6 50                add     a,80            ;calculate number of characters to clear
```

```
034E    47                  ld      b,a
034F    7D                  ld      a,l             ;save cursor column
0350    36 20       clrl:   ld      (hl),' '        ;clear next location
0352    2C                  inc     l
0353    10 FB               djnz    clrl            ;if end of line not clear
0355    6F                  ld      l,a             ;restore cursor column
0356    C9                  ret

                    ;;      clrscn - clear visible screen memory.
                    ;
0357    21 3000     clrscn: ld      hl,crtmem       ;home cursor
035A    3E 17               ld      a,23
035C    32 FFB1             ld      (base),a        ;put line 23 at bottom of screen
035F    D3 14               out     (scroll),a      ;note scroll register gets A8-A12, not d0-d7

                    ;;      clreos - clear to end of screen.
                    ;
0361    CD 0344     clreos: call    clreol          ;clear remainder of current row
0364    E5                  push    hl              ;save cursor location
0365    ED 4B FFB1  clrsl:  ld      bc,(base)       ;set bottom screen row to c
0369    7D                  ld      a,l
036A    17                  rla
036B    7C                  ld      a,h
036C    17                  rla                     ;get row# component of hl into a
036D    E6 1F               and     00011111b
036F    B9                  cp      c
0370    28 08               jr      z,clrs2         ;if hl is on bottom row of screen
0372    CD 02E7             call    dncsr           ;point hl to next row
0375    CD 0341             call    clrlin          ;and fill that line with spaces
0378    18 EB               jr      clrsl
037A    E1          clrs2:  pop     hl              ;restore original cursor pointer
037B    C9                  ret

                    ;;      lindel - Line delete.
                    ;
037C    E5          lindel: push    hl              ;save cursor address
037D    CD 040F             call    bbg             ;bury balcones gold
0380    29                  add     hl,hl
0381    7C                  ld      a,h
0382    E6 1F               and     00011111b       ;extract row
0384    ED 4B FFB0          ld      bc,(base-1)     ;get base screen row in b
0388    CD 03D1     lind1:  call    smp             ;set move parameters
038B    B8                  cp      b
038C    28 10               jr      z,lind2         ;if last line
038E    C5                  push    bc              ;b=last line, c=row
038F    01 0050             ld      bc,80
0392    ED B0               ldir
0394    C1                  pop     bc
0395    79                  ld      a,c
0396    3C                  inc     a
0397    FE 18               cp      24
0399    38 ED               jr      c,lind1
039B    AF                  xor     a               ;wrap
039C    18 EA               jr      lind1           ;move next line
039E    EB          lind2:  ex      de,hl
039F    CD 0341     lind3:  call    clrlin
```

```
03A2    E1                      pop     hl              ;restore original cursor
03A3    C9                      ret

                        ;;      linins - Line insert.
                        ;
03A4    E5              linins: push    hl              ;save cursor position
03A5    3E 17                   ld      a,23            ;get bottom line
03A7    CD F31E                 call    cca             ;compute cursor address
03AA    CD 040F                 call    bbg             ;bury balcones gold
03AD    E1                      pop     hl
03AE    E5                      push    hl
03AF    29                      add     hl,hl
03B0    7C                      ld      a,h
03B1    E6 1F                   and     00011111b       ;extract cursor row
03B3    47                      ld      b,a
03B4    3A FFB1                 ld      a,(base)        ;set last line
03B7    B8              linil:  cp      b
03B8    28 13                   jr      z,lini3         ;if move complete
03BA    3D                      dec     a
03BB    F2 03C0                 jp      p,lini2
03BE    3E 17                   ld      a,23
03C0    CD 03D1         lini2:  call    smp             ;set move parameters
03C3    EB                      ex      de,hl
03C4    C5                      push    bc
03C5    01 0050                 ld      bc,80
03C8    ED B0                   ldir
03CA    C1                      pop     bc
03CB    18 EA                   jr      linil           ;move next line
03CD    E1              lini3:  pop     hl
03CE    E5                      push    hl              ;restore cursor
03CF    18 CE                   jr      lind3           ;clear cursor line

                        ;;      smp - Set move parameters.
                        ;
03D1    4F              smp:    ld      c,a             ;save row
03D2    CD F324                 call    cca2
03D5    EB                      ex      de,hl
03D6    79                      ld      a,c
03D7    CD F323                 call    cca1
03DA    79                      ld      a,c
03DB    C9                      ret

                        ;;      chrins - Character insert.
                        ;
03DC    E5              chrins: push    hl
03DD    7D                      ld      a,l             ;set cursor column
03DE    E6 7F                   and     01111111b       ;set move length = 79-column
03E0    ED 44                   neg
03E2    C6 4F                   add     a,79
03E4    47                      ld      b,a             ;number of chars to move
03E5    7E                      ld      a,(hl)          ;get char under cursor
03E6    36 20                   ld      (hl),' '        ;clear char under cursor
03E8    28 06                   jr      z,chrin2        ;if cursor in last column
03EA    2C              chrin1: inc     l
03EB    4E                      ld      c,(hl)
03EC    77                      ld      (hl),a
```

```
03ED    79                      ld      a,c
03EE    10 FA                   djnz    chrin1
03F0    32 F319         chrin2: ld      (gold),a        ;shift line into the gold mine
03F3    E1                      pop     hl
03F4    C9                      ret

                        ;;      chrdel - Character delete.
                        ;
03F5    E5              chrdel: push    hl
03F6    7D                      ld      a,l
03F7    E6 7F                   and     01111111b       ;isolate cursor column
03F9    ED 44                   neg
03FB    C6 4F                   add     a,79
03FD    4F                      ld      c,a             ;number of chars to move = 79-column
03FE    06 00                   ld      b,0
0400    54                      ld      d,h
0401    5D                      ld      e,l
0402    1A                      ld      a,(de)
0403    32 F319                 ld      (gold),a        ;mine balcones gold
0406    23                      inc     hl
0407    C4 0418                 call    nz,ldirx
040A    EB                      ex      de,hl
040B    36 20                   ld      (hl),' '        ;blank last char on line
040D    E1                      pop     hl              ;restore cursor
040E    C9                      ret

                        ;;      bbg - bury balcones gold.
                        ;
040F    CD 02F2         bbg:    call    return
0412    11 FF5C                 ld      de,linbuf
0415    01 0050                 ld      bc,80
0418    ED B0           ldirx:  ldir
041A    C9                      ret

                                subttl  Logical to Physical Driver Executioner
                                page
```

```
Balcones Operating System for the XEROX 820-II  MACRO-80 3.44   09-Dec-81
Logical to Physical Driver Executioner

                                       above                   ;code goes in ram
        0518!                +         d&seg

                               ;;      Xqdvr - Execute Physical Driver.
                               ;
                               ;       Entry:  HL = Pointer to Physical Drive Request Block
                               ;
                               ;
        F344     23            xqdvr:  inc     hl              ;point at physical unit
        F345     E5                    push    hl
        F346     23                    inc     hl
        F347     5E                    ld      e,(hl)          ;set logical drive
        F348     21 F360               ld      hl,Seltab       ;Set driver mapping table address
        F34B     16 00                 ld      d,0
        F34D     19                    add     hl,de           ;index into driver select table
        F34E     19                    add     hl,de
        F34F     5E                    ld      e,(hl)          ;set physical driver index
        F350     23                    inc     hl
        F351     7E                    ld      a,(hl)          ;set physical unit
        F352     21 F380               ld      hl,Drvtab       ;set Driver table address
        F355     19                    add     hl,de
        F356     19                    add     hl,de
        F357     5E                    ld      e,(hl)          ;set physical driver address
        F358     23                    inc     hl
        F359     56                    ld      d,(hl)
        F35A     E1                    pop     hl              ;recover request block address
        F35B     77                    ld      (hl),a          ;store physical unit
        F35C     2B                    dec     hl
        F35D     D5                    push    de              ;execute physical driver
        F35E     C9                    ret

                                       subttl  Physical Disk Driver Area
                                       page
```

```

                                    ;;      Physical Disk Driver Area.
                                    ;
                                            above
        035F"               +               d&seg
                                    ;;      Waste space to get drivers on page boundry.
                                    ;
        F35F                        Wasted:
        000F                        bndry   equ     0fh
                                            if      ($ and bndry) ne 0
        F35F                                defs    bndry+1-($ and bndry),-1
                                            endif

                                    ;;      Logical to Physical Drive Mapping Tables.
                                    ;
                                    ;       Seltab contains two bytes per logical CP/M drive A-P.
                                    ;       The first byte is an index into the physical driver
                                    ;       address table below.  The second byte is a unit number
                                    ;       that is  passed to the driver by the standard deblocker.
                                    ;
        F360                        Seltab:

        F360    01 00               .A:     defb    1,0             ;Floppy Unit 0
        F362    01 01               .B:     defb    1,1             ;Floppy Unit 1
        F364    01 02               .C:     defb    1,2             ;Floppy Unit 2
        F366    01 03               .D:     defb    1,3             ;Floppy Unit 3

        F368    01 04               .E:     defb    1,4             ;Rigid Partition 0
        F36A    01 05               .F:     defb    1,5             ;Rigid Partition 1
        F36C    01 06               .G:     defb    1,6             ;Rigid Partition 2
        F36E    01 07               .H:     defb    1,7             ;Rigid Partition 3

        F370    00 00               .I:     defb    0,0             ;Error Driver
        F372    00 00               .J:     defb    0,0             ;Error Driver
        F374    00 00               .K:     defb    0,0             ;Error Driver
        F376    00 00               .L:     defb    0,0             ;Error Driver

        F378    00 00               .M:     defb    0,0             ;Error Driver
        F37A    00 00               .N:     defb    0,0             ;Error Driver
        F37C    00 00               .O:     defb    0,0             ;Error Driver
        F37E    00 00               .P:     defb    0,0             ;Error Driver

                                    ;;      Physical Driver Address Table.
                                    ;
                                    ;       Dvrtab contains the addresses of several independent
                                    ;       physical disk drivers.  By convention, driver # 0 always
                                    ;       returns a select error.
                                    ;
        F380    F42A                Drvtab: defw    Selerr          ;select error physical driver
        F382    F4B0                        defw    Dskdvr          ;Disk driver (WD or SA)
        F384    0000                        defw    0               ;Empty physical driver expansion slots
        F386    0000                        defw    0
        F388    0000                        defw    0
        F38A    0000                        defw    0
        F38C    0000                        defw    0
```

```
F38E    FFFF                    defw    0-1             ;mark last empty expansion slot

                        ;;      Overlaid Monitor Ram Address Definitions.
                        ;
FC80                    dirbuf  equu    0fc80h          ;director buffer
FD00                    chk00   equ     0fd00h          ;Directory Check Vector for Floppy Drive 0
FD20                    chk01   equ     0fd20h          ;Directory Check Vector for Floppy Drive 1
FD40                    chk02   equ     0fd40h          ;Directory Check Vector for Floppy Drive 2
FD60                    chk03   equ     0fd60h          ;Directory Check Vector for Floppy Drive 3
0000                    chk04   equ     0               ;No Check Vector for Rigid Partition 0
0000                    chk05   equ     0               ;No Check Vector for Rigid Partition 1
0000                    chk06   equ     0               ;No Check Vector for Rigid Partition 2
0000                    chk07   equ     0               ;No Check Vector for Rigid Partition 3

FD80                    all00   equ     0fd80h          ;Floppy Drive 0 Allocation Vector
FDA0                    all01   equ     0fda0h          ;Floppy Drive 1 Allocation Vector
FDC0                    all02   equ     0fdc0h          ;Floppy Drive 2 Allocation Vector
FDE0                    all03   equ     0fde0h          ;Floppy Drive 3 Allocation Vector
FE00                    all04   equ     0fe00h          ;Rigid Partition Allocation vectors
FE80                    all05   equ     0fe80h
FEC0                    all06   equ     0fec0h
FEE0                    all07   equ     0fee0h

                                subttl  Disk Parameter Headers
                                page
```

```

                                  ;;      Disk Parameter Headers.
                                  ;
        F390    0000 0000         Dpbase: dw      0,0,0,0         ;Floppy Drive 0
        F394    0000 0000
        F398    FC80 0000                 dw      dirbuf,0
        F39C    FD00 FD80                 dw      chk00,al100

        F3A0    0000 0000                 dw      0,0,0,0         ;Floppy Drive 1
        F3A4    0000 0000
        F3A8    FC80 0000                 dw      dirbuf,0
        F3AC    FD20 FDA0                 dw      chk01,al101

        F3B0    0000 0000                 dw      0,0,0,0         ;Floppy Drive 2
        F3B4    0000 0000
        F3B8    FC80 0000                 dw      dirbuf,0
        F3BC    FD40 FDC0                 dw      chk02,al102

        F3C0    0000 0000                 dw      0,0,0,0         ;Floppy Drive 3
        F3C4    0000 0000
        F3C8    FC80 0000                 dw      dirbuf,0
        F3CC    FD60 FDE0                 dw      chk03,al103

        F3D0    0000 0000                 dw      0,0,0,0         ;Rigid Partition 0
        F3D4    0000 0000
        F3D8    FC80 F470                 dw      dirbuf,Dpbrg4
        F3DC    0000 FE00                 dw      chk04,al104

        F3E0    0000 0000                 dw      0,0,0,0         ;Rigid Partition 1
        F3E4    0000 0000
        F3E8    FC80 F480                 dw      dirbuf,Dpbrg5
        F3EC    0000 FE80                 dw      chk05,al105

        F3F0    0000 0000                 dw      0,0,0,0         ;Rigid Partition 2
        F3F4    0000 0000
        F3F8    FC80 F490                 dw      dirbuf,Dpbrg6
        F3FC    0000 FEC0                 dw      chk06,al106

        F400    0000 0000                 dw      0,0,0,0         ;Rigid Partition 3
        F404    0000 0000
        F408    FC80 F4A0                 dw      dirbuf,Dpbrg7
        F40C    0000 FEE0                 dw      chk07,al107

                                          subttl  Sector Translate Tables
                                          page
```

```

                                 ;;      Sector Translation Tables.
                                 ;
                                 ;       For 8 inch single density drives.
                                 ;       Skew by 6
                                 ;
F410     01 07 0D 13             trn6:   db      01,07,13,19
F414     19 05 0B 11                     db      25,05,11,17
F418     17 03 09 0F                     db      23,03,09,15
F41C     15 02 08 0E                     db      21,02,08,14
F420     14 1A 06 0C                     db      20,26,06,12
F424     12 18 04 0A                     db      18,24,04,10
F428     10 16                           db      16,22

F42A     21 0000                 selerr: ld      hl,0
F42D     F6 FF                           or      -1
F42F     C9                              ret

                                         subttl  Floppy Disk Parameter Blocks
                                         page
```

```
                                ;;      Floppy Disk Parameter Blocks, one per media format.
                                ;
       F430                     dpb8s:
                                ;       Single Density, Single side

       F430    001A                     dw      26              ;spt
       F432    03 07 00                 db      3,7,0           ;blkshf, blkmsk, nullmsk
       F435    00F2 003F                dw      242,63,192,16,2 ;dsw,dirm,alloc01,chksiz,trk off
       F439    00C0 0010
       F43D    0002
       F43F    00                       db      0               ;128 byte sectors

                                ;       Single Density, Double Side

       F440    001A                     dw      26              ;spt
       F442    04 0F 01                 db      4,15,1          ;blkshf, blkmsk, nullmsk
       F445    00F6 007F                dw      246,127,192,16,2;dsw,dirm,alloc01,chksiz,trk off
       F449    00C0 0010
       F44D    0002
       F44F    00                       db      0               ;128 byte sectors

       F450                     dpb8d:
                                ;       Double Density, Single Side

       F450    0034                     dw      2*26            ;spt
       F452    04 0F 01                 db      4,15,1          ;blkshf, blkmsk, nullmsk
       F455    00F2 007F                dw      242,127,192,32,2;dsw,dirm,alloc01,chksiz,trk off
       F459    00C0 0020
       F45D    0002
       F45F    81                       db      81h             ;256 byte sectors, track zero single density

                                ;       Double Density, Double Side

       F460    0034                     dw      2*26            ;spt
       F462    05 1F 03                 db      5,31,3          ;blkshf, blkmsk, nullmsk
       F465    00F6 007F                dw      246,127,192,32,2;dsw,dirm,alloc01,chksiz,trk off
       F469    00C0 0020
       F46D    0002
       F46F    81                       db      81h             ;256 byte sectors, track zero single density

                                        subttl  Micro Floppy Disk Parameter Blocks
                                        page
```

```

F470                            dpb5s:

                                ;       Single Density, Single Side

F470    0012                            dw      18              ;spt
F472    03 07 00                        db      3,7,0           ;blkshf, blkmsk, nulmsk
F475    0052 001F                       dw      82,31,128,8,3   ;dsw,dirm,alloc01,chksiz,trk off
F479    0080 0008
F47D    0003
F47F    00                              db      0               ;128 byte sectors

                                ;       Single Density, Double Side

F480    0012                            dw      18              ;spt
F482    03 07 00                        db      3,7,0           ;blkshf, blkmsk, nulmsk
F485    00AC 001F                       dw      172,31,128,8,3  ;dsw,dirm,alloc01,chksiz,trk off
F489    0080 0008
F48D    0003
F48F    00                              db      0               ;128 byte sectors

F490                            dpb5d:

                                ;       Double Density, Single Side

F490    0022                            dw      17*2            ;spt
F492    03 07 00                        db      3,7,0           ;blkshf, blkmsk, nulmsk
F495    009C 003F                       dw      156,63,192,16,3 ;dsw,dirm,alloc01,chksiz,trk off
F499    00C0 0010
F49D    0003
F49F    81                              db      81h             ;256 byte sectors, track zero single density

                                ;       Double Density, Double Side

F4A0    0022                            dw      17*2            ;spt
F4A2    04 0F 01                        db      4,15,1          ;blkshf, blkmsk, nulmsk
F4A5    00A2 003F                       dw      162,63,192,16,3 ;dsw,dirm,alloc01,chksiz,trk off
F4A9    00C0 0010
F4AD    0003
F4AF    81                              db      81h             ;256 byte sectors, track zero single density

                                        subttl  Western Digital WD-1797-02 Floppy Disk Driver
                                        page
```

```
                                        ;;      Standard Disk Driver Interface Definitions.
                                        ;
                                        ;       The main entry point (FLOPPY) is called with HL pointing
                                        ;       to a disk driver request block.  All information is passed
                                        ;       in this request as follows:
                                        ;
                                        ;       HL->    db      command ;1 = read, 0 = write, -1 = select dph
                                        ;               db      phunit  ;physical unit for request (0-3)
                                        ;               db      cpunit  ;CP/M logical drive for request (0-15)
                                        ;               dw      track   ;CP/M track number (offset already applied)
                                        ;               dw      sector  ;Phys sector number (after deblocking)
                                        ;               dw      address ;CP/M dma transfer address
                                        ;
                                                subttl  Assembly Constants
                                                page
```

Balcones Operating System for the XEROX 820-II MACRO-80 3.44 09-Dec-81
Assembly Constants

```
0066            NMI     equ     00066h          ;address of non maskable interrupt
                ;;      WD 1797 I/O port addresses.
                ;
0010            wdsr    equ     10h             ;status
0010            wdcr    equ     10h             ;command
0011            wdtr    equ     11h             ;track
0012            wdsn    equ     12h             ;sector
0013            wddt    equ     13h             ;data
001C            wdsl    equ     1ch             ;drive select port
0030            wdsd    equ     30h             ;select single density
0031            wddd    equ     31h             ;select double density
                ;;      External Disk Parameter Tables.
                ;
0007            fm.un   equ     7
0004            fm.ds   equ     4
0005            fm.dd   equ     5
0006            fm.fv   equ     6
00A0            fm.ddss equ     (1 shl fm.dd) or (1 shl fm.un)
004D            ntrk8   equ     77
0028            ntrk5   equ     40
0004            c.8in   equ     4
0005            c.two   equ     5
0006            timou   equ     6               ;motor / select time out
000A            dpbofs  equ     10              ;offset in dph for dpb address
                        subttl  Floppy Disk Driver Proper
                        page
```

```
                                  ;;      Western Digital Floppy Disk Driver.
                                  ;
      F4B0    CD F647             Dskdvr: call    rdc             ;un hang busy controller
      F4B3    7E                          ld      a,(hl)          ;set command
      F4B4    23                          inc     hl              ;point to unit
      F4B5    32 F4E7                     ld      (rdop),a
      F4B8    3C                          inc     a
      F4B9    28 55                       jr      z,selec         ;if select command
      F4BB    06 0A                       ld      b,10            ;set retry count
      F4BD    C5                  flop1:  push    bc              ;save count
      F4BE    E5                          push    hl              ;save command
      F4BF    7E                          ld      a,(hl)          ;set unit select
      F4C0    CD F544                     call    selunt
      F4C3    FA F506                     jp      m,flop5         ;if unit not ready
      F4C6    23                          inc     hl
      F4C7    23                          inc     hl
      F4C8    4E                          ld      c,(hl)          ;set track low
      F4C9    CD F5A3                     call    seekx           ;position disk
      F4CC    4E                          ld      c,(hl)          ;retrieve track low
      F4CD    20 37                       jr      nz,flop5        ;if unrecoverable error
      F4CF    23                          inc     hl              ;track high
      F4D0    23                          inc     hl
      F4D1    13                          inc     de              ;point to second byte of track table entry
      F4D2    1A                          ld      a,(de)          ;get diskette type
      F4D3    E6 18                       and     18h
      F4D5    7E                          ld      a,(hl)          ;sector low
      F4D6    20 06                       jr      nz,flop2        ;if single density, cp/m skews
      F4D8    79                          ld      a,c             ;get current logical track
      F4D9    B7                          or      a
      F4DA    7E                          ld      a,(hl)          ;set sector
      F4DB    28 01                       jr      z,flop2         ;if single density track zero
      F4DD    3C                          inc     a               ;translate for double density
      F4DE    D3 12               flop2:  out     (wdsn),a        ;set sector to read in 1791
      F4E0    23                          inc     hl              ;skip sector high
      F4E1    23                          inc     hl              ;dmal
      F4E2    5E                          ld      e,(hl)          ;set transfer address to HL
      F4E3    23                          inc     hl              ;dmah
      F4E4    56                          ld      d,(hl)
      F4E5    EB                          ex      de,hl
      F4E6    3E 00                       ld      a,0             ;set read/write switch
      F4E7                        rdop    equ     $-1
      F4E8    B7                          or      a
      F4E9    0E A8                       ld      c,0a8h          ;preset write command
      F4EB    3E A3                       ld      a,0a3h          ;set second part of OUTI
      F4ED    28 03                       jr      z,flop3         ;if write
      F4EF    0E 88                       ld      c,088h          ;turn write command into read command
      F4F1    3D                          dec     a               ;turn OUTI into INI
      F4F2    32 F4FE             flop3:  ld      (rdwra),a       ;set up i/o direction
      F4F5    3E 00                       ld      a,0
      F4F6                        rdwrs   equ     $-1             ;set side compare flag
      F4F7    81                          add     a,c
      F4F8    4F                          ld      c,a
      F4F9    CD F61D                     call    stc             ;start transfer
      F4FC    76                  flop4:  halt                    ;wait for DRQ or INT
```

```
F4FD    ED A2                   ini                     ;transfer next byte
F4FE                    rdwra   equ     $-1
F4FF    20 FB                   jr      nz,flop4        ;if transfer not complete
F501    CD F639                 call    ttc             ;terminate transfer command
F504    E6 DF                   and     0dfh            ;set error bits
F506    E1              flop5:  pop     hl              ;recover command pointer
F507    C1                      pop     bc
F508    C8                      ret     z               ;if no errors
F509    D8                      ret     c               ;if 5.25" not ready
F50A    CD F069                 call    softv
F50D    10 AE                   djnz    flop1           ;if retry not exceeded
F50F    C9                      ret

                        ;;      select - select dph for unit.
                        ;
F510    7E              selec:  ld      a,(hl)          ;set unit
F511    FE 04                   cp      4
F513    D2 F42A                 jp      nc,selerr       ;if bad unit select
F516    26 00                   ld      h,0
F518    CD F039                 call    dayti           ;set address of timers
F51B    2B                      dec     hl              ;point to motor timer
F51C    22 F559                 ld      (mtradr),hl
F51F    2B                      dec     hl              ;point to configurable step rate
F520    22 F66E                 ld      (stpadr),hl     ;store address of step rate for media selector
F523    26 00                   ld      h,0
F525    6F                      ld      l,a
F526    29                      add     hl,hl           ;multiply by 2**4
F527    29                      add     hl,hl
F528    29                      add     hl,hl
F529    29                      add     hl,hl
F52A    11 F390                 ld      de,Dpbase       ;set address of disk parameter headers
F52D    19                      add     hl,de           ;set dph address
F52E    E5                      push    hl
F52F    CD F65A                 call    smf             ;set media format
F532    E1                      pop     hl
F533    CA F42A                 jp      z,selerr        ;if no media
F536    71                      ld      (hl),c          ;store translate table
F537    23                      inc     hl
F538    70                      ld      (hl),b
F539    01 000A                 ld      bc,dpbofs
F53C    09                      add     hl,bc           ;point to dpb addr and clear carry
F53D    72                      ld      (hl),d          ;fill in dpb address
F53E    2B                      dec     hl
F53F    73                      ld      (hl),e
F540    ED 42                   sbc     hl,bc           ;point back to dph
F542    AF                      xor     a
F543    C9                      ret

                        ;;      select physical unit.
                        ;
F544    4F              selunt: ld      c,a             ;save select
F545    EB                      ex      de,hl
F546    21 F700                 ld      hl,trktbl+1     ;set track / density table address
F549    06 00                   ld      b,0
F54B    09                      add     hl,bc
F54C    09                      add     hl,bc
```

```
F54D    22 F6D1             ld      (smfa),hl       ;save address for media selector
F550    EB                  ex      de,hl
F551    1A                  ld      a,(de)          ;get previous density switch
F552    1B                  dec     de              ;point to track word
F553    CD F598             call    selden          ;select density
F556    3E 06               ld      a,timou         ;start motor / select timer
F558    32 F559             ld      (mtradr),a
F559            mtradr      equ     $-2             ;address filled in by once only routine
F55B    FB                  ei                      ;insure clock enabled
F55C    CB 89               res     1,c             ;map C->A, D->B
F55E    DB 1C               in      a,(wdsl)        ;read current select
F560    47                  ld      b,a
F561    E6 F8               and     not 7
F563    B1                  or      c               ;insert new select
F564    3C                  inc     a               ;0-1, 1-2
F565    D3 1C               out     (wdsl),a        ;select drive
F567    A8                  xor     b
F568    E6 03               and     3
F56A    28 25               jr      z,sel3          ;if drive select identical
F56C    3E FF               ld      a,-1            ;force track position recovery
F56E    12                  ld      (de),a
F56F    CB 60               bit     c.8in,b         ;test 8/5 status
F571    20 1E               jr      nz,sel3         ;if 8"
F573    CD F647             call    rdc             ;set type I status
F576    06 08               ld      b,2*4           ;watch for four holes (8 transitions)
F578    E5      sellw:      push    hl
F579    2A F559             ld      hl,(mtradr)     ;get address of motor select timer
F57C    7E                  ld      a,(hl)
F57D    E1                  pop     hl
F57E    D6 04               sub     timou-2         ;look for 1-2 seconds
F580    D8                  ret     c               ;if drive not spinning
F581    DB 10               in      a,(wdsr)
F583    E6 02               and     2
F585    28 F1   sel2:       jr      z,sellw         ;if index not under light
F587    3A F585             ld      a,(sel2)        ;switch index polarity
F58A    EE 08               xor     8               ;(jr z) xor (jr nz)
F58C    32 F585             ld      (sel2),a
F58F    10 E7               djnz    sellw           ;wait for at least three revolutions
F591    DB 10   sel3:       in      a,(wdsr)        ;set ready status
F593    E6 80               and     80h
F595    C9                  ret

F596    3E 18   seldns:     ld      a,18h           ;set track zero single density
F598    32 F632 selden:     ld      (dsw),a         ;store switch for read/write routines
F59B    E6 18               and     18h
F59D    D3 31               out     (wddd),a        ;pre-select dual density
F59F    C8                  ret     z               ;if dual density
F5A0    D3 30               out     (wdsd),a        ;select single density
F5A2    C9                  ret

                ;;          seek - position disk.
                ;
F5A3    79      seekx:      ld      a,c             ;set new track
F5A4    B7                  or      a
F5A5    CC F596             call    z,seldns        ;force single density track 0
F5A8    13                  inc     de
```

```
F5A9    1A                  ld      a,(de)
F5AA    1B                  dec     de
F5AB    E6 01               and     1
F5AD    28 15               jr      z,sek1          ;if one sided diskette
F5AF    DB 1C               in      a,(wdsl)
F5B1    CB 67               bit     c.8in,a
F5B3    06 4D               ld      b,ntrk8         ;set number of eight inch tracks
F5B5    20 02               jr      nz,sek0         ;if 8" drives
F5B7    06 28               ld      b,ntrk5
F5B9    79          sek0:   ld      a,c             ;set seek track
F5BA    B8                  cp      b
F5BB    3E 00               ld      a,0             ;preset side 0
F5BD    38 05               jr      c,sek1          ;if side 0
F5BF    79                  ld      a,c
F5C0    90                  sub     b               ;wrap to side 1
F5C1    4F                  ld      c,a
F5C2    3E 02               ld      a,2             ;set side 1
F5C4    32 F4F6     sek1:   ld      (rdwrs),a       ;store F1 (update SSO)
F5C7    87                  add     a,a             ;move into select port position
F5C8    47                  ld      b,a
F5C9    F3                  di
F5CA    DB 1C               in      a,(wdsl)
F5CC    CB 97               res     2,a
F5CE    B0                  or      b
F5CF    FB                  ei
F5D0    D3 1C               out     (wdsl),a        ;send out REAL SSO
F5D2    1A                  ld      a,(de)          ;check current position
F5D3    D3 11               out     (wdtr),a        ;inform 1797 of current track
F5D5    B9                  cp      c
F5D6    28 17               jr      z,seek3         ;if position ok, load head
F5D8    3C                  inc     a               ;check for forced recovery
F5D9    CC F5F8             call    z,rse           ;recover seek errors
F5DC    28 0D               jr      z,seek1         ;if error not recoverable
F5DE    79          seek0:  ld      a,c             ;set new track
F5DF    D3 13               out     (wddt),a        ;in data register
F5E1    3E 1C               ld      a,1ch           ;set seek with verify command
F5E3    CD F643             call    isc             ;issue step command
F5E6    E6 98               and     98h
F5E8    79                  ld      a,c             ;update current track
F5E9    28 02               jr      z,seek2         ;if no errors
F5EB    F6 FF       seek1:  or      -1              ;force recovery next time
F5ED    12          seek2:  ld      (de),a
F5EE    C9                  ret
F5EF    CD F647     seek3:  call    rdc             ;set type I status
F5F2    E6 20               and     20h             ;test head load
F5F4    28 E8               jr      z,seek0         ;if head is not loaded
F5F6    AF          retzr:  xor     a               ;say seek complete
F5F7    C9                  ret

                    ;;      rse - recover seek error.
                    ;
F5F8    C5          rse:    push    bc
F5F9    CD F605             call    rdid            ;read id mark
F5FC    20 05               jr      nz,rse1         ;if track position identified
F5FE    CD F641             call    recal           ;recalibrate
F601    E6 04               and     4               ;verify track zero flag set
```

```
F603    C1              rsel:   pop     bc
F604    C9                      ret

                        ;;      rdid - read id mark.
                        ;
F605    0E C4           rdid:   ld      c,0c4h          ;set Read Address Command
F607    CD F61D                 call    stc             ;start transfer command
F60A    76                      halt                    ;wait for interrupt
F60B    ED 40                   in      b,(c)           ;first byte is track
F60D    76                      halt
F60E    ED 48                   in      c,(c)           ;second byte is side, pitch next 4
F610    CD F639                 call    ttc             ;terminate transfer command
F613    E6 98                   and     98h             ;ignore lost data
F615    20 04                   jr      nz,rdid1        ;if track not identified
F617    78                      ld      a,b
F618    D3 11                   out     (wdtr),a        ;tell 1797 track head is on now
F61A    F6                      defb    0f6h            ;or xra to set NZ
F61B    AF              rdid1:  xor     a               ;set track not found
F61C    C9                      ret

                        ;;      stc - start transfer command.
                        ;
F61D    F3              stc:    di                      ;lock normal interrupts
F61E    3A 0066                 ld      a,(NMI)         ;save byte at NMI address
F621    32 F63A                 ld      (ttca),a
F624    3E C9                   ld      a,0c9h          ;store RET there
F626    32 0066                 ld      (NMI),a
F629    79                      ld      a,c             ;retrieve command
F62A    01 1413                 ld      bc,wddt+20*256  ;1797 access timer / data port
F62D    D3 10                   out     (wdcr),a        ;issue command
F62F    10 FE                   djnz    $               ;pause 60 usec
F631    3E 00                   ld      a,0
F632                    dsw     equ     $-1             ;density switch
F633    E6 18                   and     18h             ;say ready and density
F635    C8                      ret     z               ;if double density
F636    06 80                   ld      b,128           ;set 128 byte single density sectors
F638    C9                      ret

                        ;;      ttc - terminate transfer command.
                        ;
F639    3E 00           ttc:    ld      a,0             ;restore location 66
F63A                    ttca    equ     $-1
F63B    32 0066                 ld      (NMI),a
F63E    FB                      ei                      ;take interrupts now
F63F    18 0A                   jr      woc             ;wait for 1797 to complete

                        ;;      recalibrate drive.
                        ;
F641    AF              recal:  xor     a               ;set restore command / track 0
F642    12                      ld      (de),a          ;set track zero

                        ;;      isc - issue step command.
                        ;
F643    F6 01           isc:    or      1               ;insert step rate
F644                    stepr   equ     $-1     ;*****=>;modify here for step rate change
F645    18 02                   jr      icc
```

```
Balcones Operating System for the XEROX 820-II  MACRO-80 3.44   09-Dec-81
Floppy Disk Driver Proper

F647    3E D0               rdc:    ld      a,0d0h          ;terminate and set type I status
                            ;;      icc  - issue controller command.
                            ;
F649    D3 10               icc:    out     (wdcr),a        ;issue 1797 command
                            ;;      woc  - wait operation complete.
                            ;
F64B    3E 14               woc:    ld      a,20            ;set 60 usec delay
F64D    3D                  woc1:   dec     a
F64E    20 FD                       jr      nz,woc1
F650    CD F066             woc2:   call    idle            ;idle cpu
F653    DB 10                       in      a,(wdsr)        ;set 1797 status
F655    CB 47                       bit     0,a
F657    20 F7                       jr      nz,woc2         ;if busy, wait
F659    C9                          ret
                                    subttl  Media Format Selector
                                    page
```

```



                           ;;      smf - set media format.
                           ;
                           ;       entry:  unit in A
                           ;
                           ;       exit:   DE = dpb address
                           ;               BC = translate table
                           ;
F65A    CD F544            smf:    call    selunt          ;select unit
F65D    FA F5F6                    jp      m,retzr         ;if disk not ready
F660    21 F6D5                    ld      hl,dtype
F663    36 A0                      ld      (hl),fm.ddss    ;start out double density, single side, retry
F665    DB 1C                      in      a,(wdsl)        ;read select status
F667    CB 67                      bit     c.8in,a         ;test 8" / 5" status
F669    20 02                      jr      nz,smf0         ;if 8 inch drives
F66B    CB F6                      set     fm.fv,(hl)      ;move up to 5.25" dpbs
F66D    3A F66E            smf0:   ld      a,(stpadr)      ;set configurable step rate for 8" drives
F66E                       stpadr  equ     $-2
F670    E6 03                      and     3               ;just so seeks aren't formats
F672    32 F644                    ld      (stepr),a       ;save step rate in seek command
F675    CD F641            smf0a:  call    recal           ;establish position
F678    E6 84                      and     84h
F67A    C8                         ret     z               ;if not on track zero
F67B    FA F5F6                    jp      m,retzr         ;if unit not ready
F67E    D3 31                      out     (wddd),a        ;set double density
F680    3E FF                      ld      a,-1
F682    12                         ld      (de),a          ;clear drive on track
F683    3E 02                      ld      a,2             ;use track 2 for density select
F685    D3 13                      out     (wddt),a
F687    3E 18                      ld      a,18h           ;seek / no verify
F689    CD F643                    call    isc             ;issue seek command
F68C    3E 1C                      ld      a,1ch           ;find id mark
F68E    D3 10                      out     (wdcr),a        ;start verify
F690    01 0000                    ld      bc,0            ;set timers
F693    10 FE              smf1:   djnz    $               ;pause
F695    DB 10                      in      a,(wdsr)
F697    CB 47                      bit     0,a
F699    28 08                      jr      z,smf1a         ;if command completed
F69B    0D                         dec     c
F69C    20 F5                      jr      nz,smf1         ;if more time
F69E    CD F647                    call    rdc             ;terminate seek
F6A1    3E 18                      ld      a,18h           ;set pseudo record not found
F6A3    E6 18              smf1a:  and     18h             ;check record not found / crc error
F6A5    13                         inc     de              ;point to density word in track table
F6A6    12                         ld      (de),a
F6A7    1B                         dec     de
F6A8    28 14                      jr      z,smf2          ;if density select successful
F6AA    D3 30                      out     (wdsd),a        ;use single density
F6AC    3E 1C                      ld      a,1ch           ;verify single density
F6AE    CD F643                    call    isc             ;issue seek
F6B1    E6 18                      and     18h
F6B3    28 07                      jr      z,smf1b         ;if single density successful
F6B5    CB 7E                      bit     fm.un,(hl)      ;test retry
F6B7    CB BE                      res     fm.un,(hl)      ;clear retry
```

```
F6B9    20 BA                       jr      nz,smf0a        ;if retry
F6BB    C9                          ret                     ;return select error
F6BC    CB AE               smf1b:  res     fm.dd,(hl)      ;back up to single density dpbs
F6BE    CB BE               smf2:   res     fm.un,(hl)      ;clear retry
F6C0    DB 1C                       in      a,(wdsl)
F6C2    CB D7                       set     2,a             ;select side 2
F6C4    D3 1C                       out     (wdsl),a
F6C6    CD F605                     call    rdid            ;read id mark
F6C9    28 09                       jr      z,smf4          ;if no id found, must be one side
F6CB    0D                          dec     c
F6CC    20 06                       jr      nz,smf4         ;if side 1 ID not read
F6CE    CB E6                       set     fm.ds,(hl)      ;bump up to two sided dpbs
F6D0    21 F6D1                     ld      hl,smfa         ;set double sided status in track table
F6D1                        smfa    equ     $-2
F6D3    34                          inc     (hl)
F6D4    21 0000             smf4:   ld      hl,0            ;set diskette type
F6D5                        dtype   equ     $-2
F6D7    7D                          ld      a,l             ;save type
F6D8    4C                          ld      c,h             ;preset no translate
F6D9    44                          ld      b,h
F6DA    11 F430                     ld      de,dpb8s        ;set base of disk parameter blocks
F6DD    19                          add     hl,de
F6DE    EB                          ex      de,hl           ;return DPB address in DE
F6DF    CB 6F                       bit     fm.dd,a
F6E1    C0                          ret     nz              ;if diskette is double density
F6E2    01 F6ED                     ld      bc,trn5         ;preset 5.25" skew table
F6E5    CB 77                       bit     fm.fv,a
F6E7    C0                          ret     nz              ;if diskette is small
F6E8    01 F410                     ld      bc,trn6         ;set 8" translate
F6EB    3C                          inc     a               ;force NZ
F6EC    C9                          ret

                            ;;      Skew by 5 translate table.
                            ;
F6ED    01 06 0B 10         trn5:   db      01,06,11,16
F6F1    03 08 0D 12                 db      03,08,13,18
F6F5    05 0A 0F 02                 db      05,10,15,02
F6F9    07 0C 11 04                 db      07,12,17,04
F6FD    09 0E                       db      09,14

F6FF    7F 00 C0 00         trktbl: db      7fh,0,0c0h,0,20h,0,2,0,81h
F703    20 00 02 00
F707    81

F708                        rigdpb  equ     0f708h
F770                        iobloc  equ     0f770h

                                    above
0708"               +               d&seg

                                    .dephase
                                    .phase  0f470h
F470                        sasstr  equ     $

                                    Subttl  Rigid Partition Disk Parameter Blocks.
                                    page
```

```

                                ;       There are 16 4k Blocks per pseudo track.
                                ;       Track Zero (2 Cylinders) are reserved for CP/M boot.
                                ;
     0040                       Nt4     equ     64              ;Number of Tracks on Partition 0
     0020                       Nt5     equ     32              ;Number of Tracks on Partition 1
     0010                       Nt6     equ     16              ;Number of Tracks on Partition 2
     0010                       Nt7     equ     16              ;Number of Tracks on Partition 3

     0000                       ..      aset    0               ;First usable track
                                        irpc    n,<4567>
                                ..      aset    ..+1            ;reserve system track
                                Dsm&n   equ     Nt&n*16-17
                                Rtk&n   equ     ..
                                ..      aset    ..+Nt&n-1
                                Dpbrg&n:dw      512             ;spt
                                        db      5,31            ;blkshf, blkmsk
                                        db      3+2*(Dsm&n ge 256);exm
                                        dw      Dsm&n           ;dsm
                                        dw      511             ;dirmax
                                        db      -1              ;alloc0 (reserve additional dir space)
                                        db      0               ;alloc1
                                        dw      0               ;check size
                                        dw      Rtk&n           ;track offset
                                        db      1               ;256 byte sectors
                                        endm
     F470    0200           +   Dpbrg&4:dw      512             ;spt
     F472    05 1F          +           db      5,31            ;blkshf, blkmsk
     F474    01             +           db      3+2*(Dsm&4 ge 256);exm
     F475    03EF           +           dw      Dsm&4           ;dsm
     F477    01FF           +           dw      511             ;dirmax
     F479    FF             +           db      -1              ;alloc0 (reserve additional dir space)
     F47A    00             +           db      0               ;alloc1
     F47B    0000           +           dw      0               ;check size
     F47D    0001           +           dw      Rtk&4           ;track offset
     F47F    01             +           db      1               ;256 byte sectors
     F480    0200           +   Dpbrg&5:dw      512             ;spt
     F482    05 1F          +           db      5,31            ;blkshf, blkmsk
     F484    01             +           db      3+2*(Dsm&5 ge 256);exm
     F485    01EF           +           dw      Dsm&5           ;dsm
     F487    01FF           +           dw      511             ;dirmax
     F489    FF             +           db      -1              ;alloc0 (reserve additional dir space)
     F48A    00             +           db      0               ;alloc1
     F48B    0000           +           dw      0               ;check size
     F48D    0041           +           dw      Rtk&5           ;track offset
     F48F    01             +           db      1               ;256 byte sectors
     F490    0200           +   Dpbrg&6:dw      512             ;spt
     F492    05 1F          +           db      5,31            ;blkshf, blkmsk
     F494    03             +           db      3+2*(Dsm&6 ge 256);exm
     F495    00EF           +           dw      Dsm&6           ;dsm
     F497    01FF           +           dw      511             ;dirmax
     F499    FF             +           db      -1              ;alloc0 (reserve additional dir space)
     F49A    00             +           db      0               ;alloc1
     F49B    0000           +           dw      0               ;check size
     F49D    0061           +           dw      Rtk&6           ;track offset
```

```
        F49F    01              +               db      1               ;256 byte sectors
        F4A0    0200            +       Dpbrg&7:dw      512             ;spt
        F4A2    05 1F           +               db      5,31            ;blkshf, blkmsk
        F4A4    03              +               db      3+2*(Dsm&7 ge 256);exm
        F4A5    00EF            +               dw      Dsm&7           ;dsm
        F4A7    01FF            +               dw      511             ;dirmax
        F4A9    FF              +               db      -1              ;alloc0 (reserve additional dir space)
        F4AA    00              +               db      0               ;alloc1
        F4AB    0000            +               dw      0               ;check size
        F4AD    0071            +               dw      Rtk&7           ;track offset
        F4AF    01              +               db      1               ;256 byte sectors

                                                subttl  SA1403 - Shugart / DTC SASI Driver
                                                page
```

```


                    ;;      SA-1403D SASI driver.
                    ;
 EE00               rgdbuf  equ     0ee00h          ;rigid parameter load buffer

                    ;;      Sasi Pio Port Addresses.
                    ;
 0011               pioAs   equ     11h             ;Pio A Status
 0010               pioAd   equ     pioAs xor 01b
 0013               pioBs   equ     pioAs xor 10b
 0012               pioBd   equ     pioAs xor 11b

 0010               Sasid   equ     pioAd           ;bus data
 0012               Sasic   equ     pioBd           ;bus control
 0012               Sasis   equ     pioBd           ;bus status

 001C               syspio  equ     1ch             ;system configuration port

                    ;;      Sasi controller status bit definitions.
                    ;
 0000               b.bsy   equ     00              ;(in)  controller busy status
 0001               b.msg   equ     01              ;(in)  status byte completion status
 0002               b.cd    equ     02              ;(in)  control byte or data byte transfer
 0003               b.req   equ     03              ;(in)  controller request for data/command
 0004               b.io    equ     04              ;(in)  data transfer direction
 0005               b.sel   equ     05              ;(out) controller select
 0006               b.par   equ     06              ;(in)  buss parity error
 0007               b.rst   equ     07              ;(out) controller reset

                    ;;      Logical Unit Assignments.
                    ;
 0000               falun   equ     0               ;A: Lun
 0001               fblun   equ     1               ;B: Lun
 0000               fclun   equ     0               ;C: Lun
 0002               fdlun   equ     2               ;D: Lun
 0003               rglun   equ     3               ;E: Lun

                            subttl  Sasi Class Code Definitions
                            page
```

```
Balcones Operating System for the XEROX 820-II  MACRO-80 3.44   09-Dec-81
Sasi Class Code Definitions

                        ;;      Class Command Codes for Prom Set AS31*
                        ;
                        ;       DTC Reference Manual Dated February 4, 1981.
                        ;
                        ;;      class  0 commands.
                        ;
        0000            c.trdy  equ     00h             ;test ready status
        0001            c.recal equ     01h             ;recalibrate drive
        0002            c.rsyn  equ     02h             ;request syndrome
        0003            c.rqsn  equ     03h             ;request sense after error
        0004            c.fmat  equ     04h             ;format drive
        0005            c.vtrk  equ     05h             ;verify track format
        0006            c.ftrk  equ     06h             ;format single track
        0007            c.flaw  equ     07h             ;format track with flaw
        0008            c.read  equ     08h             ;read data
        0009            c.wrpr  equ     09h             ;write protect sector
        000A            c.writ  equ     0ah             ;write data
        000B            c.seek  equ     0bh             ;initiate seek
        000C            c.init  equ     0ch             ;inititialize drive

                        ;;      Class 6 commands.
                        ;
        00C0            c.flpy  equ     0c0h            ;define floppy disk format

                        ;;      Floppy Format Codes.
                        ;
        0000            fmds    equ     0               ;double side bit
        0001            fmdd    equ     1               ;double density bit
        0002            fm.sz   equ     2               ;sector size bit
        0003            fm.wr   equ     3               ;log2(fm.ddds+1)

        0000            fm.sdss equ     00h             ;Single Density, Single Sided
        0001            fm.sdds equ     01h             ;Single Density, Double Sided
        0006            fmddss  equ     06h             ;Double Density, Single Sided
        0007            fm.ddds equ     07h             ;Double Density, Double Sided
        0080            fm.hard equ     80h             ;Rigid

                        ;;      Class 7 commands.
                        ;
        00E0            c.tram  equ     0e0h            ;test ram buffer

                        ;;      Message Macros.
                        ;
                        pmsg    macro   n,msg
                                if1
                                .printx +MSG N+
                                endif
                                endm

                        phex    macro   n,m
                                .radix  16
                                pmsg    %(n),<m>
                                .radix  10
                                endm
```

```

                                        subttl  Sasi Physical Driver.
                                        page
```

```
                                   ;;      Sa1403 - Sasi Physical Driver.
                                   ;

        F4B0    06 06              Sa1403: ld      b,6             ;set retry count
        F4B2    7E                 sas0a:  ld      a,(hl)          ;set driver operation
        F4B3    23                         inc     hl              ;point to unit
        F4B4    3C                         inc     a
        F4B5    28 49                      jr      z,sselec        ;if select DPH
        F4B7    E5                         push    hl
        F4B8    C5                         push    bc
        F4B9    3D                         dec     a
        F4BA    3E 0A                      ld      a,c.writ        ;preset write opcode
        F4BC    28 02                      jr      z,sas0
        F4BE    3E 08                      ld      a,c.read        ;assume read
        F4C0    32 F6F0            sas0:   ld      (opcode),a      ;set Sasi opcode
        F4C3    7E                         ld      a,(hl)          ;get driver unit
        F4C4    E5                         push    hl
        F4C5    CD F5AF                    call    mlu             ;map to logical unit
        F4C8    E1                         pop     hl
        F4C9    23                         inc     hl              ;ignore cpm dsk
        F4CA    23                         inc     hl              ;track low
        F4CB    56                         ld      d,(hl)          ;set track
        F4CC    23                         inc     hl
        F4CD    23                         inc     hl
        F4CE    5E                         ld      e,(hl)          ;set sector
        F4CF    CD F6D7                    call    cwp             ;check write protect
        F4D2    20 22                      jr      nz,sas2         ;if write protected and track > 0
        F4D4    E5                         push    hl              ;save request block address
        F4D5    CD F5C2                    call    mpa             ;map physical address to logical address
        F4D8    21 F6F0                    ld      hl,opcode
        F4DB    CD F643                    call    iccs            ;issue controller command
        F4DE    E1                         pop     hl              ;get pointer to low sector back
        F4DF    23                         inc     hl              ;ignore sector high
        F4E0    23                         inc     hl              ;dma low
        F4E1    5E                         ld      e,(hl)
        F4E2    23                         inc     hl              ;dma high
        F4E3    56                         ld      d,(hl)
        F4E4    EB                         ex      de,hl           ;set transfer address to HL
        F4E5    06 00                      ld      b,0             ;set sector length
        F4E6                       seclen  equ     $-1
        F4E7    3A F6F0                    ld      a,(opcode)
        F4EA    FE 0A                      cp      c.writ
        F4EC    28 05                      jr      z,sas1          ;if write command
        F4EE    CD F65F                    call    tdi             ;transfer data in
        F4F1    18 03                      jr      sas2
        F4F3    CD F656            sas1:   call    tdo             ;transfer data out
        F4F6    C1                 sas2:   pop     bc
        F4F7    E1                         pop     hl
        F4F8    2B                         dec     hl
        F4F9    C8                         ret     z               ;if no errors
        F4FA    CD F069                    call    softv           ;report soft error
        F4FD    10 B3                      djnz    sas0a           ;if more retries
        F4FF    C9                         ret                     ;return error
```

```
                                        subttl  Physical Driver Select
                                        page
```

```
                                ;;      Select - Physical Driver Select.
                                ;
F500    7E                      sselec: ld      a,(hl)          ;set physical unit
F501    FE 08                           cp      8               ;verify in range
F502                            numunt  equ     $-1
F503    3F                              ccf
F504    D4 F50E                         call    nc,smfs         ;set media format
F507    D0                              ret     nc              ;if media identified

                                ;;      selerr - Select Error Driver.
                                ;
F508    21 0000                 xselerr: ld     hl,0            ;Select Error Driver
F50B    F6 FF                   selerl: or      -1
F50D    C9                              ret

                                ;;      smfs - Set Media Format.
                                ;
                                ;       entry:  A = Driver unit index
                                ;       Exit:   HL = DPH address, if no carry
                                ;
F50E    CD F6F6                 smfs:   call    first           ;execute first time only routine
F511    CD F5AF                         call    mlu             ;map to logical unit
F514    EB                              ex      de,hl           ;get dph index to hl
F515    7D                              ld      a,l             ;and A
F516    29                              add     hl,hl           ;index *16
F517    29                              add     hl,hl
F518    29                              add     hl,hl
F519    29                              add     hl,hl
F51A    11 F390                         ld      de,Dpbase       ;set base of Disk Parameter Headers
F51D    19                              add     hl,de
F51E    FE 04                           cp      4
F520    D0                              ret     nc              ;if rigid unit
F521    E5                              push    hl              ;save dph address
F522    3E 80                           ld      a,80h           ;disable error recovery
F524    32 F6F5                         ld      (dctrl),a
F527    32 F5F1                         ld      (lastfm+1),a
F52A    0A                              ld      a,(bc)          ;always try double side first
F52B    F6 01                           or      1 shl fmds
F52D    02                              ld      (bc),a
F52E    3E 07                           ld      a,8-1           ;try each type two times
F530    32 F53D                 smfs1:  ld      (smfsa),a       ;set retry count
F533    C5                              push    bc              ;save define format table address
F534    CD F57A                         call    cdd             ;check drive density
F537    C1                              pop     bc
F538    60                              ld      h,b             ;set format table address
F539    69                              ld      l,c
F53A    28 14                           jr      z,smfs2         ;if diskette type identified
F53C    3E 00                           ld      a,0
F53D                            smfsa   equ     $-1             ;diskette type retry counter
F53E    D6 01                           sub     1
F540    38 31                           jr      c,smfs4         ;if media not identified
F542    35                              dec     (hl)            ;advance disk type code
F543    F2 F548                         jp      p,smfs1a        ;if no wrap
F546    36 07                           ld      (hl),fm.ddds
```

```
F548    CB 4E           smfs1a: bit     fmdd,(hl)       ;test for double density
F54A    20 E4                   jr      nz,smfs1        ;if double density
F54C    CB 96                   res     fm.sz,(hl)      ;set sector size = 128
F54E    18 E0                   jr      smfs1           ;try more diskette types
F550    57              smfs2:  ld      d,a             ;preset no translate
F551    5A                      ld      e,d
F552    CB 4E                   bit     fmdd,(hl)
F554    20 03                   jr      nz,smfs3        ;if diskette is double density
F556    11 F410                 ld      de,trn6         ;set single density translate
F559    E1              smfs3:  pop     hl              ;get dph address
F55A    E5                      push    hl
F55B    73                      ld      (hl),e          ;store translate address
F55C    23                      inc     hl
F55D    72                      ld      (hl),d
F55E    11 0009                 ld      de,10-1
F561    19                      add     hl,de           ;point to dpb address in dph
F562    0A                      ld      a,(bc)          ;get selected format
F563    E6 03                   and     3
F565    EB                      ex      de,hl
F566    6F                      ld      l,a
F567    29                      add     hl,hl           ;index by 16
F568    29                      add     hl,hl
F569    29                      add     hl,hl
F56A    29                      add     hl,hl
F56B    01 F430                 ld      bc,dpb8s        ;set dpb base
F56E    09                      add     hl,bc           ;set dpb address (clears carry)
F56F    EB                      ex      de,hl           ;recover dpb pointer address in dph
F570    73                      ld      (hl),e
F571    23                      inc     hl
F572    72                      ld      (hl),d
F573    E1              smfs4:  pop     hl              ;get dph address
F574    3E 00                   ld      a,0             ;enable error recovery
F576    32 F6F5                 ld      (dctrl),a
F579    C9                      ret

                        ;;      cdd - check drive density.
                        ;
F57A    0A              cdd:    ld      a,(bc)          ;get attempted side
F57B    E6 01                   and     1               ;try side 1 on ds, 0 on ss
F57D    11 0201                 ld      de,2*256+1
F580    28 02                   jr      z,cdd0          ;if single side
F582    16 4F                   ld      d,77+2          ;use back side
F584    CD F5C2         cdd0:   call    mpa             ;map physical address
F587    21 F6F0                 ld      hl,opcode
F58A    36 08                   ld      (hl),c.read
F58C    CD F643                 call    iccs            ;issue controller command
F58F    CD F6CE                 call    sim             ;set input mode
F592    CD F687         cdd1:   call    wfr             ;wait for req
F595    20 04                   jr      nz,cdd2         ;if timeout or status, not data requested
F597    ED 78                   in      a,(c)           ;eat sector
F599    18 F7                   jr      cdd1
F59B    CD F669         cdd2:   call    wcc             ;wait command complete
F59E    C9                      ret

                        ;;      p2l - Physical to Logical Mapping Table.
                        ;
```

```
                             ;       First byte is the SA1403 Logical Unit Number
                             ;       Second byte is extended drive format code
                             ;
F59F    00 00                p21:    db      falun shl 5,fm.sdss     ;Floppy Drive 0
F5A1    20 01                        db      fblun shl 5,fm.sdds     ;Floppy Drive 1
F5A3    00 06                        db      falun shl 5,fmddss      ;Floppy Drive 0
F5A5    40 07                        db      fdlun shl 5,fm.ddds     ;Floppy Drive 2
F5A7    60 80                        db      rglun shl 5,fm.hard     ;Rigid Partition 0
F5A9    60 80                        db      rglun shl 5,fm.hard     ;Rigid Partition 1
F5AB    60 80                        db      rglun shl 5,fm.hard     ;Rigid Partition 2
F5AD    60 80                        db      rglun shl 5,fm.hard     ;Rigid Partition 3

                             ;;      Mlu - Map Logical Unit.
                             ;
                             ;       Entry:  A  =  Physical Driver Unit from Deblocker
                             ;       Exit:   DE =  dph index
                             ;               BC = pointer to format code
                             ;               Sasi LUN stored in command block
                             ;
F5AF    21 F59F              mlu:    ld      hl,p21          ;set Rigid Physical to Logical map table
F5B2    16 00                        ld      d,0
F5B4    5F                           ld      e,a
F5B5    19                           add     hl,de           ;point to table entry
F5B6    19                           add     hl,de
F5B7    7E                           ld      a,(hl)          ;get Sasi LUN
F5B8    32 F6F1                      ld      (lun),a         ;store in read/write command
F5BB    32 F6EB                      ld      (deflun),a      ;store in define command
F5BE    23                           inc     hl              ;point to format descriptor
F5BF    44                           ld      b,h
F5C0    4D                           ld      c,l
F5C1    C9                           ret

                             ;;      Mpa - Map Physical Address.
                             ;
                             ;       Entry:  BC = p21 format code address
                             ;               D  = Track
                             ;               E  = Sector
                             ;
                             ;       Exit:   Logical Address set in command block
                             ;               Seclen set for transfer
                             ;               Controller notified of floppy format
                             ;
F5C2    21 F4E6              mpa:    ld      hl,seclen       ;preset sector length
F5C5    36 00                        ld      (hl),0
F5C7    EB                           ex      de,hl           ;preset Laddr = Paddr for rigid
F5C8    0A                           ld      a,(bc)          ;get drive format code
F5C9    FE 80                        cp      fm.hard
F5CB    28 46                        jr      z,mpa5          ;if Rigid Partition
F5CD    FE 06                        cp      fmddss
F5CF    38 06                        jr      c,mpa1          ;if single density
F5D1    2C                           inc     l               ;advance sector from 0->25 to 1->26
F5D2    7C                           ld      a,h
F5D3    B7                           or      a
F5D4    20 04                        jr      nz,mpa2         ;if not on track zero
F5D6    2D                           dec     l
F5D7    3E 80                mpa1:   ld      a,128           ;set short sector
```

```
F5D9    12                      ld      (de),a
F5DA    0A              mpa2:   ld      a,(bc)
F5DB    CB 47                   bit     fmds,a
F5DD    28 0A                   jr      z,mpa22         ;if not double sided disk
F5DF    7C                      ld      a,h             ;map first 77 tracks to side zero
F5E0    FE 4D                   cp      77
F5E2    38 02                   jr      c,mpa21         ;if side zero tracks
F5E4    D6 4D                   sub     77              ;offset to back side
F5E6    3F              mpa21:  ccf
F5E7    8F                      adc     a,a
F5E8    67                      ld      h,a
F5E9    E5              mpa22:  push    hl              ;save track/sector
F5EA    0A                      ld      a,(bc)          ;get floppy format
F5EB    2A F6EB                 ld      hl,(deflun)     ;get new unit
F5EE    67                      ld      h,a
F5EF    11 FFFF                 ld      de,-1           ;get previously used format/lun
F5F0                    lastfm  equ     $-2
F5F2    22 F5F0                 ld      (lastfm),hl     ;save this format/unit for next time
F5F5    B7                      or      a
F5F6    ED 52                   sbc     hl,de
F5F8    28 0C                   jr      z,mpa3          ;if unit and format same as last time
F5FA    32 F6EF                 ld      (flpfrm),a
F5FD    21 F6EA                 ld      hl,deflpy       ;issue define floppy command
F600    CD F643                 call    iccs
F603    CD F669                 call    wcc
F606    E1              mpa3:   pop     hl              ;recover track / sector
F607    44                      ld      b,h             ;set track
F608    11 001A                 ld      de,26           ;compute sector-26-1+(Track+1)*26
F60B    62                      ld      h,d             ;clear upper track
F60C    37                      scf
F60D    ED 52                   sbc     hl,de           ;adjust sector
F60F    04                      inc     b               ;force one pass
F610    19              mpa4:   add     hl,de           ;multiply track by sectors/track
F611    10 FD                   djnz    mpa4            ;if multiply incomplete
F613    7C              mpa5:   ld      a,h             ;swap H & L
F614    65                      ld      h,l
F615    6F                      ld      l,a
F616    22 F6F2                 ld      (addrh),hl      ;Store address in command block
F619    C9                      ret

                                subttl  Sasi Bus Control Interface
                                page
```

```
                                ;;      gca - get controller attention.
                                ;
F61A    CD F6AD                 gca:    call    reset           ;reset controller if required
F61D    7E                              ld      a,(hl)          ;get command
F61E    FE 01                           cp      c.recal
F620    3E 0A                           ld      a,9+1           ;set 9+ second time-out
F622    28 02                           jr      z,gca0          ;if recalibrate, use long time-out
F624    3E 03                           ld      a,2+1           ;use short time-out
F626    32 F627                 gca0:   ld      (gcaa),a
F627                            gcaa    equ     $-2     ;*****=>;monitor one second timer address goes here
F629    CD F6D2                         call    som             ;set output mode
F62C    3E 01                           ld      a,1             ;Set sasi controller address
F62E    D3 10                           out     (Sasid),a
F630    3E 20                           ld      a,1 shl b.sel   ;assert Select Line
F632    D3 12                           out     (Sasic),a
F634    DB 12                   gca1:   in      a,(Sasis)       ;get sasi status
F636    0F                              rrca                    ;get busy bit in C
F637    38 06                           jr      c,gca2          ;if controller is ready
F639    CD F6A5                         call    cft             ;check for time-out
F63C    F2 F634                         jp      p,gca1          ;if not timed out
F63F    AF                      gca2:   xor     a
F640    D3 12                           out     (Sasic),a       ;drop Select
F642    C9                              ret

                                ;;      iccs  - issue Controller Command.
                                ;
                                ;       HL => Sasi command block
                                ;
F643    7E                      iccs:   ld      a,(hl)          ;peek at opcode
F644    FE 04                           cp      c.fmat
F646    C8                              ret     z               ;do not allow format entire disk
F647    CD F61A                         call    gca             ;get controller attention
F64A    01 0610                         ld      bc,Sasid+6*256  ;set port / command block length
F64D    CD F687                 iccs1:  call    wfr             ;wait for REQ
F650    C8                              ret     z               ;if data requested
F651    ED A3                           outi                    ;send next byte
F653    20 F8                           jr      nz,iccs1
F655    C9                              ret

                                ;;      tdo - transmit data out.
                                ;
F656    CD F687                 tdo:    call    wfr             ;wait for req
F659    20 0E                           jr      nz,wcc          ;if not data requested
F65B    ED B3                           otir                    ;pitch sector out
F65D    18 0A                           jr      wcc

                                ;;      tdi - transmit data in.
                                ;
F65F    CD F6CE                 tdi:    call    sim             ;set input mode
F662    CD F687                         call    wfr             ;wait for req
F665    20 02                           jr      nz,wcc          ;if status, not data requested
F667    ED B2                           inir                    ;read sector

                                ;;      wcc  - wait command complete.
```

```
                                ;
F669    CD F6CE         wcc:    call    sim             ;set input mode
F66C    CD F687                 call    wfr             ;wait for REQ
F66F    28 2A                   jr      z,ecr           ;if controller not providing status
F671    ED 78                   in      a,(c)           ;read completion status
F673    E6 03                   and     3               ;ignore unused bits
F675    47                      ld      b,a
F676    CD F687                 call    wfr             ;wait for REQ
F679    28 20                   jr      z,ecr           ;if not status
F67B    DB 12                   in      a,(Sasis)       ;recover status
F67D    CB 4F                   bit     b.msg,a
F67F    28 1A                   jr      z,ecr           ;if not message byte
F681    ED 78                   in      a,(c)           ;read message byte
F683    20 16                   jr      nz,ecr          ;if last byte not zero
F685    B0                      or      b               ;set Sasi error status byte
F686    C9                      ret

                        ;;      wfr - wait for REQ.
                        ;
                        ;       Exit:   A < 0   Timer Expired
                        ;               A = 0   Request is for data
                        ;               A > 0   Request is for control
                        ;
F687    CD F6A5         wfr:    call    cft             ;check for time-out
F68A    FA F69A                 jp      m,wfr1          ;if controller hung
F68D    DB 12                   in      a,(Sasis)       ;read sasi status
F68F    CB 5F                   bit     b.req,a
F691    28 F4                   jr      z,wfr           ;if request not asserted
F693    CB 77                   bit     b.par,a         ;check buss parity
F695    20 03                   jr      nz,wfr1         ;if parity error
F696                    wfra    equ     $-1
F697    E6 04                   and     1 shl b.cd      ;test control / data bit
F699    C9                      ret
F69A    F1              wfr1:   pop     af              ;pitch return address

                        ;;      ecr - Enable Controller Reset.
                        ;
F69B    AF              ecr:    xor     a               ;enable controller reset next time
F69C    32 F6AD                 ld      (reset),a       ;by placing NOP at reset entry point
F69F    F6 FF                   or      -1              ;return error status
F6A1    32 F5F0                 ld      (lastfm),a      ;force define floppy format
F6A4    C9                      ret

                        ;;      Cft - Check for Time-out.
                        ;
F6A5    CD F066         cft:    call    idle            ;idle cpu
F6A8    3A 0000                 ld      a,(0)
F6A9                    cfta    equ     $-2     ;*****=>;This word gets the address of the timer
                                public  cfta
F6AB    B7                      or      a
F6AC    C9                      ret

                        ;;      Reset - Reset Controller.
                        ;
                        ;       This routine is called prior to every command that is
                        ;       issued to the controller, but disables itself after
```

```
                             ;       running.  When, and if the controller times-out, this
                             ;       routine is re-enabled.  Thus, the controller will be
                             ;       reset again before the next command is issued.
                             ;
F6AD     00                  reset:  nop             ;*****=>;Note RET gets put here after reset
                                                            ;     NOP gets put there if time-out
F6AE     3E CF                       ld      a,11001111b     ;initialize pio in mode 3
F6B0     D3 13                       out     (pioBs),a
F6B2     3E 5F                       ld      a,01011111b     ;d7, d5 are outputs
F6B4     D3 13                       out     (pioBs),a
F6B6     3E 80                       ld      a,1 shl b.rst   ;assert reset to controller
F6B8     D3 12                       out     (Sasic),a
F6BA     AF                          xor     a
F6BB     D3 12                       out     (Sasic),a       ;de-assert reset
F6BD     3E C9                       ld      a,0c9h
F6BF     32 F6AD                     ld      (reset),a       ;disable reset until time-out
F6C2     E5                          push    hl
F6C3     21 F6E8                     ld      hl,rgrecal
F6C6     CD F643                     call    iccs            ;issue recursive rigid recalibrate
F6C9     CD F669                     call    wcc
F6CC     E1                          pop     hl
F6CD     C9                          ret

                             ;;      Sim - Set Input Mode.
                             ;
F6CE     3E 4F               sim:    ld      a,01001111b     ;set pio A input mode
F6D0     18 02                       jr      som1

                             ;;      Som - Set Output Mode.
                             ;
F6D2     3E 0F               som:    ld      a,00001111b     ;set pio A output mode
F6D4     D3 11               som1:   out     (pioAs),a
F6D6     C9                          ret

                             ;;      cwp - check write protect.
                             ;
F6D7     0A                  cwp:    ld      a,(bc)          ;get drive type
F6D8     E6 80                       and     fm.hard
F6DA     C8                          ret     z               ;if not rigid disk access
F6DB     3E 00                       ld      a,0             ;get dirty parameter flag
F6DC                         rdonly  equ     $-1
F6DD     B7                          or      a
F6DE     C8                          ret     z               ;if not write protected
F6DF     7A                          ld      a,d
F6E0     B7                          or      a
F6E1     C8                          ret     z               ;if track zero request
F6E2     3A F6F0                     ld      a,(opcode)
F6E5     D6 08                       sub     c.read          ;allow reads, but no writes to file system
F6E7     C9                          ret

                                     subttl  Sasi Command Blocks
                                     page
```

```


                              ;;      Sasi Command Blocks.
                              ;

         F6E8   01            rgrecal:db      c.recal
         F6E9   60            reclun: db      3 shl 5

         F6EA   C0            deflpy: db      c.flpy          ;define floppy format
         F6EB   00            deflun: db      0
         F6EC   00 00 00              db      0,0,0
         F6EF   00            flpfrm: db      0

         F6F0   00            opcode: db      0               ;Class code / Operation
         F6F1   00            lun:    db      0               ;Logical Unit & Logical Address 20-16
         F6F2   00            addrh:  db      0               ;               Logical Address 15-8
         F6F3   00            addrl:  db      0               ;               Logical Address  7-0
         F6F4   01            nblk:   db      1               ;Number of Blocks
         F6F5   00            dctrl:  db      0               ;Error Retry Disable Control word

                                      subttl  Overlayable Initialization Code
                                      page
```

Balcones Operating System for the XEROX 820-II MACRO-80 3.44   09-Dec-81
Overlayable Initialization Code

```

                                     ;;      First - First time only.
                                     ;
         F6F6     F5                 first:  push    af
                                             phex    $-dskdvr,<SASI Resident Length is>
         0010                +               .radix  16
         000A                +               .radix  10
         F6F7     3E C9                      ld      a,0c9h          ;nuke self first time
         F6F9     32 F6F6                    ld      (first),a
         F6FC     26 00                      ld      h,0             ;indicate return register value
         F6FE     CD F039                    call    dayti           ;get address of monitor timer
         F701     2B                         dec     hl
         F702     22 F6A9                    ld      (cfta),hl       ;store address of timer for check routine
         F705     22 F627                    ld      (gcaa),hl       ;and for command startup

         F708     21 F767            first1: ld      hl,cnfdpb       ;point to physical driver read command
         F70B     CD F4B0                    call    sa1403          ;read partition parameters
         F70E     B7                         or      a
         F70F     CC F723                    call    z,cpb           ;if no errors then check parameter blocks
         F712     28 0D                      jr      z,first2        ;if parameters are loaded
         F714     21 F76C                    ld      hl,cnfdpb+5     ;try backup heads
         F717     7E                         ld      a,(hl)          ;get logical sector
         F718     C6 20                      add     a,32            ;advance to next head
         F71A     77                         ld      (hl),a
         F71B     20 EB                      jr      nz,first1       ;if 4 heads and 2 cylinders not attempted
         F71D     2F                         cpl                     ;set tracks > 0 read only
         F71E     32 F6DC                    ld      (rdonly),a
         F721     F1                 first2: pop     af
         F722     C9                         ret

                                     ;;      cpb - check parameter blocks.
                                     ;
         F723     21 EE00            cpb:    ld      hl,rgdbuf       ;point to dpb buffer
         F726     3A F76C                    ld      a,(cnfdpb+5)    ;get sector this dpb set came from
         F729     FE 20                      cp      32
         F72B     20 04                      jr      nz,cpb1         ;if not primary set
         F72D     7E                         ld      a,(hl)
         F72E     FE E5                      cp      0e5h
         F730     C8                         ret     z               ;use default dpbs if none configured
         F731     11 000F            cpb1:   ld      de,16-1         ;set offset from high spt to deblock control
         F734     06 04                      ld      b,4             ;verify 4 dpbs
         F736     7E                 cpb2:   ld      a,(hl)          ;set low sectors / track
         F737     B7                         or      a
         F738     C0                         ret     nz              ;if bummer sectors / track
         F739     19                         add     hl,de           ;advance to deblock control
         F73A     7E                         ld      a,(hl)
         F73B     E6 87                      and     87h
         F73D     CB 2F                      sra     a
         F73F     CB 2F                      sra     a
         F741     C0                         ret     nz              ;if bad deblocking constant
         F742     23                         inc     hl
         F743     10 F1                      djnz    cpb2
         F745     2B                         dec     hl
         F746     11 F4AF                    ld      de,Dpbrg4+16*4-1
         F749     01 0040                    ld      bc,16*4
```

```
        F74C    ED B8                     lddr                     ;move dpbs into place
        F74E    01 0010                   ld      bc,16            ;set 16 bytes/dpx
        F751    11 F3DE                   ld      de,Dpbase+4*16+14 ;point at alloc vector pointer
        F754    21 F47D                   ld      hl,Dpbrg4+13     ;point at reserved tracks
        F757    3E 04                     ld      a,4              ;count off 4 partitions
        F759    F5                cpb3:   push    af
        F75A    7E                        ld      a,(hl)           ;get reserved tracks for partition
        F75B    3D                        dec     a                ;just so nice numbers come out
        F75C    87                        add     a,a              ;16 blks/track / 8 blks/byte = 2 bytes/track
        F75D    12                        ld      (de),a           ;store low allocation vector address
        F75E    09                        add     hl,bc            ;advance to next dpb
        F75F    EB                        ex      de,hl
        F760    09                        add     hl,bc            ;advance to next dph
        F761    EB                        ex      de,hl
        F762    F1                        pop     af
        F763    3D                        dec     a
        F764    20 F3                     jr      nz,cpb3          ;if more to allocate
        F766    C8                        ret     z                ;return success

        F767    01 04 00          cnfdpb: db      01,4,0           ;read partition 0
        F76A    0000                      dw      0                ;track zero
        F76C    0020                      dw      32               ;sector 32
        F76E    EE00                      dw      rgdbuf           ;rigid parameter table buffer

        0300                      sasidl  equ     $-sasstr
                                          .dephase
        FA08                      dloc    defl    dloc+sasidl
                                          .phase  dloc

                                          above
        0A08"                 +           d&seg

        FA08                      Dvrlmt:                          ;disk driver limit

        FA08                      rqtop   equ     $                ;set required top of resident monitor

        FA08    21 0000           slerr:  ld      hl,0
        FA0B    F6 FF                     or      -1
        FA0D    C9                        ret

                                          subttl  820 Style Disk Driver Emulator
                                          page
```

```

                                  ;;      820 Style Disk Driver Emulator.
                                  ;
                                          above                   ;generate code in ram
        0A0E"             +               d&seg

        FA0E    FF                phycmd: defb    -1              ;physical Driver Command
        FA0F    FF                phyunt: defb    -1              ;physical unit
        FA10    00                phydrv: defb    00              ;logical unit
        FA11    0000              phytrk: defw    00              ;track
        FA13    0001              physec: defw    01              ;sector
        FA15    ED80              phydma: defw    bootbf          ;dma address

                                  ;;      Select - Select Unit for I/O.
                                  ;
                                  ;       Entry:  C  = Unit
                                  ;       Exit:   A  = 0  if no errors
                                  ;               A  = -1 if errors
                                  ;
        FA17    79                select: ld      a,c             ;set drive selected
        FA18    FE 08                     cp      8
        FA1A    30 1D                     jr      nc,sel1
        FA1C    32 FA10                   ld      (phydrv),a      ;save logical CP/M drive
        FA1F    21 FA5A                   ld      hl,seltbl       ;set select table address
        FA22    06 00                     ld      b,0
        FA24    09                        add     hl,bc           ;index into select table
        FA25    7E                        ld      a,(hl)
        FA26    B7                        or      a
        FA27    EB                        ex      de,hl
        FA28    67                        ld      h,a             ;in case previous select worked, say no dph
        FA29    6F                        ld      l,a             ;to internal routines
        FA2A    C8                        ret     z               ;if drive has already been selected
        FA2B    D5                        push    de              ;save table address
        FA2C    06 FF                     ld      b,-1            ;set Select operation
        FA2E    CD FA51                   call    xqphys          ;execute physical driver
        FA31    7D                        ld      a,l             ;get returned dph address
        FA32    B4                        or      h
        FA33    D1                        pop     de              ;retrieve select table address
        FA34    28 03                     jr      z,sel1          ;if select unsuccessful
        FA36    AF                        xor     a               ;return no errors
        FA37    12                        ld      (de),a          ;prevent more density re-selects
        FA38    C9                        ret
        FA39    F6 FF             sel1:   or      -1              ;return error
        FA3B    C9                        ret

                                  ;;      Home - Position to track zero.
                                  ;
        FA3C    0E 00             home:   ld      c,0             ;force track zero

                                  ;;      Seek - Seek Track.
                                  ;
                                  ;       Entry:  C  = Track to read/write from next
                                  ;
        FA3E    79                seek:   ld      a,c
        FA3F    32 FA11                   ld      (phytrk),a
```

```
FA42    AF                       xor     a               ;return no errors
FA43    C9                       ret
                         ;;      Write - Write Physical Sector.
                         ;
FA44    06 00            write:  ld      b,0             ;set Write operation
FA46    18 02                    jr      rdwr
                         ;;      Read - Read Physical Sector.
                         ;
FA48    06 01            read:   ld      b,1             ;set read operation
                         ;;      Rdwr - Read/Write Processor.
                         ;
                         ;       Entry:  C  = Sector
                         ;               HL = Transfer Address
                         ;       Exit:   A  = 0 if no errors
                         ;               A  = -1 if errors
                         ;
FA4A    79               rdwr:   ld      a,c
FA4B    32 FA13                  ld      (physec),a      ;set physical sector
FA4E    22 FA15                  ld      (phydma),hl     ;set transfer address
                         ;;      xqphys - Internal Execute Physical Driver.
                         ;
                         ;       Entry:  B  = -1 for Select
                         ;               B  =  0 for Write
                         ;               B  =  1 for Read
                         ;
FA51    21 FA0E          xqphys: ld      hl,phycmd       ;point to physical command block
FA54    70                       ld      (hl),b          ;store operation
FA55    CD F344                  call    xqdvr           ;execute driver
FA58    B7                       or      a               ;set flags
FA59    C9                       ret
                         ;;      Emulator Disk I/O Ram.
                         ;
FA5A    FF FF FF FF      seltbl: defb    -1,-1,-1,-1     ;drive already selected table
FA5E    FF FF FF FF              defb    -1,-1,-1,-1
                                 subttl  Command processor
                                 page
```

```
                                                 
                                           above                       ;put code upstairs
         0A62"                  +          d&seg
                                    ;;     prompt user for command.
                                    ;
         FA62    FB                 prompt: ei
         FA63    31 0000                   ld       sp,stack           ;reset system stack
         FA66    CD FC3D                   call     pnext
         FA69    0D 0A                     defb     cr,lf
         FA6B    2A 20                     defm     '* '
         FA6D    04                        defb     eot
         FA6E    21 FF5C                   ld       hl,linbuf
         FA71    0E 50                     ld       c,80               ;buffer of 80 chars (ver. 2.0)
         FA73    CD FB37                   call     getlin             ;input a bufered console line
         FA76    38 51                     jr       c,what             ;print 'what ?' if input error
         FA78    3A FF5C            autobt: ld      a,(linbuf)         ;get first character in line
         FA7B    FE 0D                     cp       cr
         FA7D    28 E3                     jr       z,prompt           ;jump if a null line
         FA7F    D6 40                     sub      '@'
         FA81    FE 1B                     cp       'Z'-'@'+1
         FA83    30 44                     jr       nc,what            ;if not letter
         FA85    87                        add      a,a
         FA86    4F                        ld       c,a
         FA87    06 00                     ld       b,0
         FA89    21 FAD9                   ld       hl,cmdtab+1        ;index command table with character
         FA8C    09                        add      hl,bc
         FA8D    7E                        ld       a,(hl)
         FA8E    2B                        dec      hl
         FA8F    6E                        ld       l,(hl)             ;get address of command processor
         FA90    67                        ld       h,a
         FA91    E6 80                     and      80h
         FA93    20 17                     jr       nz,prmt1           ;if resident command
         FA95    11 FC55                   ld       de,cloc            ;move transient command to RAM area
         FA98    D5                        push     de
         FA99    01 0299                   ld       bc,tpamax          ;set length of largest transient
         FA9C    F3                        di
         FA9D    DB 1C                     in       a,(syspio)
         FA9F    B7                        or       a
         FAA0    F5                        push     af
         FAA1    F4 F29C                   call     p,crton            ;enable rom if disabled
         FAA4    ED B0                     ldir
         FAA6    F1                        pop      af
         FAA7    F4 F293                   call     p,crtoff           ;disable rom if enabled
         FAAA    FB                        ei
         FAAB    E1                        pop      hl                 ;set execution address
         FAAC    E5                 prmt1:  push    hl
         FAAD    CD FC36                   call     crlf
         FAB0    FD 21 FF5D                ld       iy,linbuf+1
         FAB4    CD FB5F                   call     params             ;input numeric parameters from
         FAB7    DD E1                     pop      ix                 ;line buffer and test if error
         FAB9    2A FFB5                   ld       hl,(param1)
         FABC    ED 5B FFB7                ld       de,(param2)
         FAC0    ED 4B FFB9                ld       bc,(param3)
         FAC4    CD FAD6                   call     jpix               ;call subroutine @ ix
         FAC7    30 99                     jr       nc,prompt          ;go back to prompt if no errors
```

```
        FAC9    CD FC3D         what:   call    pnext
        FACC    20 77 68 61             defm    ' what?'
        FAD0    74 3F
        FAD2    07                      defb    'G'-64          ;say 'what ?' and squeal
        FAD3    04                      defb    eot
        FAD4    18 8C                   jr      prompt
        FAD6    DD E9           jpix:   jp      (ix)            ;call subroutine @ ix
        FAD8    177B            cmdtab: defw    help            ;@ - Help user
        FADA    11B8                    defw    boot            ;a - boot cp/m
        FADC    1353                    defw    baud            ;b - bit rate
        FADE    1436                    defw    block           ;c - memory block move
        FAE0    12F2                    defw    memdmp          ;d - dump memory in hex/ascii
        FAE2    1315                    defw    view            ;e - enter memory
        FAE4    1428                    defw    fill            ;f - fill memory
        FAE6    12DB                    defw    goto            ;g - goto program
        FAE8    14E2                    defw    term            ;h - host terminal
        FAEA    13CA                    defw    incmd           ;i - read from input port
        FAEC    FAC9                    defw    what            ;j - not used
        FAEE    FAC9                    defw    what            ;k - not used
        FAF0    11B8                    defw    boot            ;l - load system
        FAF2    1315                    defw    view            ;m - memory examine/change
        FAF4    FAC9                    defw    what            ;n - not used
        FAF6    13F1                    defw    outcmd          ;o - write to output port
        FAF8    1459                    defw    proto           ;p - printer protocol
        FAFA    FAC9                    defw    what            ;q - not used
        FAFC    1367                    defw    dskcmd          ;r - display disk sector data
        FAFE    FAC9                    defw    what            ;s - not used
        FB00    1477                    defw    type            ;t - typewriter mode
        FB02    FAC9                    defw    what            ;u - not used
        FB04    1443                    defw    vercmd          ;v - memory block compare
        FB06    1367                    defw    dskcmd          ;w - disc sector write command
        FB08    13FB                    defw    test            ;x - ram diagnostic
        FB0A    FAC9                    defw    what            ;y - not used
        FB0C    FAC9                    defw    what            ;z - not used
        0036                    cmdsiz  equ     $-cmdtab
        FB0E    BE              check:  cp      (hl)
        FB0F    C8                      ret     z               ;return if (hl)=a
        FB10    F5                      push    af
        FB11    CD FB22                 call    mdata           ;print what was actually read
        FB14    CD FC3D                 call    pnext
        FB17    73 68 6F 75             defm    'should='
        FB1B    6C 64 3D
        FB1E    04                      defb    eot
        FB1F    F1                      pop     af
        FB20    18 07                   jr      put2j
        FB22    CD FC36         mdata:  call    crlf
        FB25    CD FC16                 call    put4hs
        FB28    7E                      ld      a,(hl)
        FB29    C3 FC1B         put2j:  jp      put2hs
                                        subttl  Console support routines
```

```
 3347                                          page
```

```
                                 ;;      getlin - read line into buffer.
                                 ;
FB2C    3E 40                    gethlp: ld      a,'@'
FB2E    32 FF5C                          ld      (linbuf),a
FB31    3E 0D                            ld      a,cr
FB33    32 FF5D                          ld      (linbuf+1),a
FB36    C9                               ret
FB37    41                       getlin: ld      b,c             ;save max line length parameter in b
FB38    CD FC27                  glinl:  call    echo            ;get a character from the console
FB3B    FE 1E                            cp      Helpkey
FB3D    28 ED                            jr      z,gethlp        ;if user needs help
FB3F    77                               ld      (hl),a
FB40    FE 0D                            cp      cr              ;check for carriage return
FB42    C8                               ret     z               ;if end of line
FB43    FE 08                            cp      'H'-64          ;check for ctl-h backspace
FB45    28 09                            jr      z,glin4
FB47    FE 20                            cp      ' '
FB49    D8                               ret     c               ;other control characters are illegal
FB4A    23                               inc     hl              ;store character in buffer
FB4B    0D                               dec     c
FB4C    20 EA                            jr      nz,glinl        ;get another if there's more room
FB4E    37                               scf
FB4F    C9                               ret                     ;return with carry=1 if too
                                                                 ;many characters are entered
FB50    2B                       glin4:  dec     hl              ;delete last character from buffer
FB51    CD FC3D                          call    pnext
FB54    20 08                            defb    ' ','H'-64      ;delete character from screen
FB56    04                               defb    eot
FB57    0C                               inc     c
FB58    78                               ld      a,b             ;set max line length
FB59    91                               sub     c
FB5A    30 DC                            jr      nc,glinl        ;if backspace not past the start of the line
FB5C    C9                               ret

FB5D    FD 23                    para0:  inc     iy              ;advance character scan
FB5F    01 00FF                  params: ld      bc,low -1       ;set parameter index
FB62    FD 7E 00                         ld      a,(iy+0)        ;fetch character
FB65    D6 0D                            sub     cr
FB67    C8                               ret     z               ;if no parameters
FB68    D6 13                            sub     ' '-cr
FB6A    28 F1                            jr      z,para0         ;if leading blanks
FB6C    0C                       para1:  inc     c               ;advance parameter index
FB6D    CB 51                            bit     2,c
FB6F    37                               scf
FB70    C0                               ret     nz              ;error if > 4 numbers entered
FB71    C5                       para2:  push    bc              ;save parameter count
FB72    CD FBDA                          call    gethex          ;read a number from line buffer
FB75    C1                               pop     bc
FB76    DD 21 FFB5               para4:  ld      ix,param1       ;point to parameter storage area
FB7A    DD 09                            add     ix,bc           ;add parameter count in bc
FB7C    DD 09                            add     ix,bc
FB7E    DD 75 00                         ld      (ix+0),l
FB81    DD 74 01                         ld      (ix+1),h        ;store data returned from 'GETHEX'
FB84    FE 20                            cp      ' '
```

```
FB86    28 E4               jr      z,paral         ;get another item if space
FB88    FE 2C               cp      ','
FB8A    28 E0               jr      z,paral         ;get another item if comma
FB8C    79                  ld      a,c             ;set parameter count
FB8D    3C                  inc     a
FB8E    C9                  ret

                    ;;      dump - dump memory.
                    ;
FB8F    E5          dump:   push    hl              ;save starting address
FB90    CD FC16             call    put4hs          ;print starting address in hex
FB93    CD FC1E             call    space
FB96    06 10               ld      b,16
FB98    3E 0F       dump2:  ld      a,16-1          ;skip 3 columns on 16 byte boundry
FB9A    CD FC23             call    dmpfmt
FB9D    3E 07               ld      a,8-1           ;skip 2 columns on 8 byte boundry
FB9F    CD FC23             call    dmpfmt
FBA2    3E 03               ld      a,4-1           ;skip 1 column  on 4 byte boundry
FBA4    CD FC23             call    dmpfmt
FBA7    7E                  ld      a,(hl)          ;get a data byte @ hl
FBA8    23                  inc     hl
FBA9    CD FC1B             call    put2hs          ;print the data in hex
FBAC    10 EA               djnz    dump2           ;repeat 16 times
FBAE    CD FC1E             call    space
FBB1    E1                  pop     hl              ;restore starting address
FBB2    06 10               ld      b,16
FBB4    3E 1F       dump3:  ld      a,1fh           ;force next character
FBB6    CD F00C             call    conout
FBB9    7E                  ld      a,(hl)          ;get back data byte @ hl
FBBA    23                  inc     hl
FBBB    CD F00C             call    conout          ;print ascii character in a
FBBE    10 F4               djnz    dump3
FBC0    CD F006             call    const           ;check console status
FBC3    28 0C               jr      z,dump4         ;if char not ready
FBC5    CD F009             call    conin           ;read char
FBC8    FE 0D               cp      cr
FBCA    C8                  ret     z               ;if user abort
FBCB    CD F009             call    conin           ;pause while user examines display
FBCE    FE 0D               cp      cr
FBD0    C8                  ret     z               ;if user found it
FBD1    CD FC36     dump4:  call    crlf            ;send end of line
FBD4    1B                  dec     de
FBD5    7A                  ld      a,d
FBD6    B3                  or      e
FBD7    20 B6               jr      nz,dump         ;if dump not complete
FBD9    C9                  ret

                    ;;      gethex converts ascii to binary.
                    ;
                    ;       carry set on illegal conversion result
                    ;       terminating character returns in a.
                    ;       hl returns with 16 bit binary integer
                    ;
FBDA    21 0000     gethex: ld      hl,0            ;preset result
FBDD    54                  ld      d,h
FBDE    5D                  ld      e,l
```

```
FBDF    29              gnum1:  add     hl,hl           ;multiply result by 16
FBE0    29                      add     hl,hl
FBE1    29                      add     hl,hl
FBE2    29                      add     hl,hl
FBE3    19                      add     hl,de           ;append next digit
FBE4    FD 7E 00        gnum3:  ld      a,(iy+0)        ;get next character from line buffer
FBE7    4F                      ld      c,a
FBE8    FD 23                   inc     iy              ;advance buffer address
FBEA    CD FBF3                 call    hexbin          ;convert one ascii hex to binary
FBED    5F                      ld      e,a
FBEE    30 EF                   jr      nc,gnum1
FBF0    79                      ld      a,c             ;return first non hex digit
FBF1    B7                      or      a
FBF2    C9                      ret

                        ;;      hexbin - convert hex to binary.
                        ;
FBF3    D6 30           hexbin: sub     '0'
FBF5    D8                      ret     c
FBF6    FE 0A                   cp      10
FBF8    3F                      ccf
FBF9    D0                      ret     nc
FBFA    D6 07                   sub     7
FBFC    FE 0A                   cp      10
FBFE    D8                      ret     c
FBFF    FE 10                   cp      16
FC01    3F                      ccf
FC02    C9                      ret

FC03    F5              put2hx: push    af
FC04    1F                      rra
FC05    1F                      rra
FC06    1F                      rra
FC07    1F                      rra
FC08    CD FC0C                 call    putnib
FC0B    F1                      pop     af
FC0C    E6 0F           putnib: and     00001111b
FC0E    C6 90                   add     a,90h
FC10    27                      daa
FC11    CE 40                   adc     a,40h
FC13    27                      daa
FC14    18 0A                   jr      output

FC16    7C              put4hs: ld      a,h
FC17    CD FC03                 call    put2hx
FC1A    7D                      ld      a,l
FC1B    CD FC03         put2hs: call    put2hx

                        ;;      space - output space.
                        ;
FC1E    3E 20           space:  ld      a,' '           ;fall through to output space

FC20    C3 F00C         output: jp      conout          ;display character

                        ;;      dmpfmt - Dump Command Output Formatter.
                        ;
```

```
FC23    A5                  dmpfmt: and     1               ;check address boundry
FC24    C0                          ret     nz              ;if not on boundry
FC25    18 F7                       jr      space           ;skip one column
                            ;;      echo - read and echo console character.
                            ;
                            ;       Echo inputs one character from the console
                            ;       device, prints it on the console output and
                            ;       then returns it in the A register in upper case.
                            ;
FC27    CD F009             echo:   call    conin           ;input a character and echo it
FC2A    FE 1E                       cp      Helpkey
FC2C    C8                          ret     z               ;do not echo help key
FC2D    CD F00C                     call    conout
FC30    FE 61                       cp      'a'
FC32    D8                          ret     c               ;if not lower case
FC33    D6 20                       sub     'a'-'A'         ;convert lower case to upper case
FC35    C9                          ret
                            ;;      crlf - carriage return-linefeed.
                            ;
FC36    CD FC3D             crlf:   call    pnext           ;print next message
FC39    0D 0A 04                    defb    cr,lf,eot
FC3C    C9                          ret
                            ;;      pnext - print message after call.
                            ;
FC3D    E3                  pnext:  ex      (sp),hl         ;set message address
FC3E    7E                          ld      a,(hl)
FC3F    23                          inc     hl
FC40    E3                          ex      (sp),hl         ;set return address
FC41    FE 04                       cp      eot
FC43    C8                          ret     z
FC44    CD F00C                     call    conout
FC47    18 F4                       jr      pnext
                                    if      options and (o.move or o.verf)
                            ;;      set block address for move and verify.
                            ;
FC49    EB                  blocad: ex      de,hl
FC4A    B7                          or      a               ;clear carry
FC4B    ED 52                       sbc     hl,de           ;get diffrence between
FC4D    EB                          ex      de,hl           ;hl & de for bytecount
FC4E    D5                          push    de              ;exchange de,bc
FC4F    50                          ld      d,b
FC50    59                          ld      e,c
FC51    C1                          pop     bc
FC52    03                          inc     bc              ;get count+1 into bc
FC53    C9                          ret
                                    endif
FC54    C9                          ret
                                    subttl  Transient Command Area
                                    page
```

```
FC55                          cloc    defl    $               ;establish overlay execution address
                                      overlay start
0C55"                     +           c&seg
                              ;;      signon - Announce System Ready.
                              ;
FC55    08                    signon: ex      af,af'
FC56    CD F293                       call    crtoff          ;disable rom/ram
FC59    08                            ex      af,af'
FC5A    28 4A                         jr      z,sign4         ;if Rx1984 loaded disk driver
FC5C    08                            ex      af,af'          ;get syspio data
FC5D    21 F091                       ld      hl,confg        ;point to configuration byte
FC60    CB 47                         bit     0,a
FC62    28 24                         jr      z,sign3         ;if SASI interface present
FC64    F3                            di
FC65    3E CF                         ld      a,11001111b     ;set Pio B in Bit Mode
FC67    D3 1D                         out     (sysctl),a
FC69    3E 38                         ld      a,00111000b     ;turn around d0,1,2
FC6B    D3 1D                         out     (sysctl),a
FC6D    D3 1C                         out     (syspio),a      ;drop all drive selects
FC6F    3E D0                         ld      a,0d0h          ;reset wd-1797-02
FC71    D3 10                         out     (wd1797),a
FC73    10 FE                 sign1:  djnz    sign1           ;wait 1797 not busy
FC75    DB 1C                         in      a,(syspio)
FC77    CB 67                         bit     c.five,a
FC79    3E 02                         ld      a,2             ;preset 10 msec step rate
FC7B    20 04                         jr      nz,sign2        ;if not 5"
FC7D    CB E6                         set     c.five,(hl)
FC7F    3E 03                         ld      a,3             ;set long step
FC81    D3 10                 sign2:  out     (wd1797),a      ;restore / unload heads
FC83    32 FF54                       ld      (steprt),a
FC86    18 1E                         jr      sign4
FC88    CB F6                 sign3:  set     c.sasi,(hl)     ;set Sasi card installed
FC8A    21 F708                       ld      hl,Rigdpb       ;set address of rigid dpb
FC8D    11 F470                       ld      de,dpb5s        ;set address of 5.25" floppy dpb
FC90    01 0300                       ld      bc,Sasidl       ;set sasi driver length
FC93    ED B0                         ldir                    ;Move driver down
FC95    E6 02                         and     2
FC97    20 0D                         jr      nz,sign4        ;if not A/E swap
FC99    21 F361                       ld      hl,Seltab+1
FC9C    06 08                         ld      b,8
FC9E    7E                    sign3a: ld      a,(hl)
FC9F    EE 04                         xor     4
FCA1    77                            ld      (hl),a
FCA2    23                            inc     hl
FCA3    23                            inc     hl
FCA4    10 F8                         djnz    sign3a

FCA6    CD FC3D               sign4:  call    pnext
FCA9    1A                            defb    clrs            ;clear screen
FCAA    1B 38                         defb    esc,'8'         ;set low light as default mode
FCAC    38 32 30 2D                   defm    '820-II v '
FCB0    49 49 20 76
FCB4    20
FCB5    34 2E 30 31                   defb    rev/100+'0','.',(rev mod 100)/10+'0',(rev mod 10)+'0'
```

```
FCB9    20 1F 1C 20                 defm    ' ',31,28,' 1982 Xerox Corp'
FCBD    31 39 38 32
FCC1    20 58 65 72
FCC5    6F 78 20 43
FCC9    6F 72 70
FCCC    0D 0A                       defb    cr,lf
FCCE    0A                          defb    lf
FCCF    4C 20 2D 20                 defm    'L - Load System'
FCD3    4C 6F 61 64
FCD7    20 53 79 73
FCDB    74 65 6D
FCDE    0D 0A                       defb    cr,lf

                                    if      options and o.term
FCE0    48 20 2D 20                 defm    'H - Host Terminal'
FCE4    48 6F 73 74
FCE8    20 54 65 72
FCEC    6D 69 6E 61
FCF0    6C
FCF1    0D 0A                       defb    cr,lf
                                    endif
                                    if      options and o.type
FCF3    54 20 2D 20                 defm    'T - Typewriter'
FCF7    54 79 70 65
FCFB    77 72 69 74
FCFF    65 72
FD01    0D 0A                       defb    cr,lf
                                    endif
FD03    07 04                       defb    7,eot

FD05    CD F006             eatkey: call    const
FD08    CA F003                     jp      z,warm          ;go enter monitor
FD0B    CD F009                     call    conin
FD0E    18 F5                       jr      eatkey

                                    subttl  I/O byte Drivers
                                    page
```

```

        00BB'                  +        overlay iobdvr
                                        c&seg

                                        .dephase
                                        .phase  iobloc

                                ;;      comins - Communications input status.
                                ;
        F770    DB 06           comins: in      a,(siocpa)
        F772    0F                      rrca
        F773    9F                      sbc     a,a
        F774    C9                      ret

                                ;;      cominp - Communications input data.
                                ;
        F775    DB 06           cominp: in      a,(siocpa)
        F777    0F                      rrca
        F778    30 FB                   jr      nc,cominp
        F77A    DB 04                   in      a,(siodpa)
        F77C    C3 F0E2                 jp      kbmask

                                ;;      comout - Communications output.
                                ;
        F77F    CD F788         comout: call    comots
        F782    28 FB                   jr      z,comout
        F784    79                      ld      a,c
        F785    D3 04                   out     (siodpa),a
        F787    C9                      ret

                                ;;      comots - Communications output status.
                                ;
        F788    DB 06           comots: in      a,(siocpa)
        F78A    E6 04                   and     4
        F78C    C8                      ret     z
        F78D    F6 FF                   or      -1
        F78F    C9                      ret

                                ;;      coniob - get console i/o byte.
                                ;
        F790    3A 0003         coniob: ld      a,(iobyte)
        F793    E6 03                   and     00000011b
        F795    C9                      ret

                                ;;      iocono - Console output through iobyte.
                                ;
        F796    CD F790         iocono: call    coniob
        F799    28 E4                   jr      z,comout
        F79B    3D                      dec     a
        F79C    CA F2FE                 jp      z,fastcrt
        F79F    79                      ld      a,c
        F7A0    C3 F0F8                 jp      sioout

                                ;;      iocons - Console status through iobyte.
                                ;
```

```
F7A3    CD F790         iocons: call    coniob
F7A6    28 C8                   jr      z,comins
F7A8    3D                      dec     a
F7A9    CA F0CD                 jp      z,kbdst
F7AC    C3 F0E5                 jp      siost

                        ;;      ioconi - Console input through iobyte.
                        ;
F7AF    CD F790         ioconi: call    coniob
F7B2    28 C1                   jr      z,cominp
F7B4    3D                      dec     a
F7B5    CA F0D8                 jp      z,kbdin
F7B8    C3 F0F0                 jp      sioin

                        ;;      lstout - List output through iobyte.
                        ;
F7BB    3A 0003         iolist: ld      a,(iobyte)
F7BE    E6 C0                   and     11000000b
F7C0    28 BD                   jr      z,comout
F7C2    EA F7DC                 jp      pe,pioout
F7C5    79                      ld      a,c
F7C6    FA F0F8                 jp      m,sioout
F7C9    C3 F2FE                 jp      fastcrt

                        ;;      List output through iobyte
                        ;
F7CC    3A 0003         iolsts: ld      a,(iobyte)
F7CF    E6 C0                   and     11000000b
F7D1    28 B5                   jr      z,comots
F7D3    EA F7F4                 jp      pe,piosto
F7D6    FA F105                 jp      m,siordy
F7D9    F6 FF                   or      -1
F7DB    C9                      ret

                        ;;      Parallel Output Driver.
                        ;
F7DC    CD F7F4         pioout: call    piosto
F7DF    28 FB                   jr      z,pioout        ;if printer not ready
F7E1    79                      ld      a,c
F7E2    D3 08                   out     (gpioda),a      ;load character data
F7E4    DB 0A                   in      a,(gpiodb)
F7E6    CB 97                   res     p.strb,a        ;assert strobe
F7E8    D3 0A                   out     (gpiodb),a
F7EA    CB D7                   set     p.strb,a        ;release stobe
F7EC    D3 0A                   out     (gpiodb),a
F7EE    3E 0A                   ld      a,10            ;delay for ACK
F7F0    3D              piol:   dec     a
F7F1    20 FD                   jr      nz,piol
F7F3    C9                      ret

                        ;;      Parallel Output Status.
                        ;
F7F4    DB 0A           piosto: in      a,(gpiodb)      ;read status
F7F6    2F                      cpl
F7F7    E6 10                   and     1 shl p.rdyo
F7F9    C8                      ret     z               ;if ready
```

```
        F7FA    F6 FF                   or      -1
        F7FC    C9                      ret

        008D                    iobdvs  equ     $-iobloc

                                        .dephase
                                        .phase  cloc+iobdvs

                                        subttl  Transient Command Processors
                                        page
```

```
                                          if      options and o.ddvr
                                ;         -- disk boot loader command --
                                ;
                                          overlay boot
0148'                  +                  c&seg

FC55     21 FF5D                          ld      hl,linbuf+1
FC58     7E                     boot1:    ld      a,(hl)           ;scan command line
FC59     2C                               inc     l
FC5A     D6 0D                            sub     cr
FC5C     28 0B                            jr      z,boot2          ;if no parameter, boot from A:
FC5E     FE 13                            cp      ' '-cr
FC60     28 F6                            jr      z,boot1          ;skip leading blanks
FC62     D6 34                            sub     'A'-cr
FC64     D8                               ret     c                ;if invalid drive
FC65     FE 10                            cp      16
FC67     3F                               ccf
FC68     D8                               ret     c                ;if bad drive
FC69     4F                     boot2:    ld      c,a              ;set boot drive selected
FC6A     C6 41                            add     a,'A'
FC6C     32 FCDD                          ld      (bootd),a        ;set up error message
FC6F     2E 00                            ld      l,0              ;set A:
FC71     C5                               push    bc
FC72     E5                               push    hl
FC73     CD FCEE                          call    swap             ;switch boot drive with A:
FC76     21 FCD9                          ld      hl,booter        ;set boot error return
FC79     E5                               push    hl
FC7A     0E 00                            ld      c,0              ;then boot from A:
FC7C     CD FA17                          call    select
FC7F     C0                               ret     nz               ;if drive not configured or density error
FC80     3E FF                            ld      a,-1
FC82     12                               ld      (de),a
FC83     11 000A                          ld      de,10            ;set dpb address offset within dph
FC86     19                               add     hl,de
FC87     5E                               ld      e,(hl)           ;set dpb address
FC88     23                               inc     hl
FC89     56                               ld      d,(hl)
FC8A     CD FA3C                          call    home
FC8D     0E 01                            ld      c,1              ;set sector 1
FC8F     1A                               ld      a,(de)           ;get low sectors per track
FC90     32 FCD4                          ld      (boots),a        ;inform boot loader
FC93     B7                               or      a
FC94     20 0D                            jr      nz,boot3         ;if not rigid
FC96     21 000D                          ld      hl,13            ;set reserved track offset within dpb
FC99     19                               add     hl,de
FC9A     4E                               ld      c,(hl)           ;get reserved tracks
FC9B     23                               inc     hl
FC9C     46                               ld      b,(hl)
FC9D     0B                               dec     bc               ;point behind directory
FC9E     ED 43 FA11                       ld      (phytrk),bc      ;do implied seek
FCA2     4F                               ld      c,a              ;set sector zero for rigid
FCA3     21 ED80                boot3:    ld      hl,bootbf        ;point to boot load buffer
FCA6     CD FA48                          call    read             ;read cold start loader
FCA9     C0                               ret     nz               ;if read error
```

```
FCAA    11 00FF                 ld      de,bootld+128-1 ;set boot loader address
FCAD    21 EDFF                 ld      hl,bootbf+128-1
FCB0    01 0080                 ld      bc,128
FCB3    ED B8                   lddr                    ;move front half of boot loader to 80h
FCB5    13                      inc     de
FCB6    1A                      ld      a,(de)          ;verify instructions read in
FCB7    FE E5                   cp      0e5h
FCB9    C8                      ret     z               ;if disk has no system
FCBA    21 F000                 ld      hl,Monitr
FCBD    22 F004                 ld      (warm+1),hl     ;set warm start to reload monitor
FCC0    CD FD05                 call    lcp             ;load configuration parameters
FCC3    21 112B                 ld      hl,iobdvr       ;load iobyte driver
FCC6    11 F770                 ld      de,iobloc
FCC9    01 008D                 ld      bc,iobdvs
FCCC    AF                      xor     a
FCCD    CD F2A3                 call    crtldir
FCD0    21 0080                 ld      hl,bootld       ;set start address
FCD3    3E 00                   ld      a,0
FCD4                    boots   equ     $-1             ;sectors per track
FCD5    11 FA0E                 ld      de,phycmd       ;tell boot loader from whence he came
FCD8    E9                      jp      (hl)            ;execute Boot Loader with return to booter

                        ;;      Booter - Boot Error Processor.
                        ;
FCD9    CD FC3D         booter: call    pnext
FCDC    07                      defb    7
FCDD    64 3A 20 4C     bootd:  defm    'd: Load error.'
FCE1    6F 61 64 20
FCE5    65 72 72 6F
FCE9    72 2E
FCEB    04                      defb    eot
FCEC    C1                      pop     bc              ;switch drives back
FCED    E1                      pop     hl

                        ;;      swap - swap logical drives.
                        ;
                        ;       Entry:  C = first  drive index, 0-15
                        ;               L = second drive index, 0-15
                        ;
FCEE    06 00           swap:   ld      b,0             ;clear upper indices
FCF0    60                      ld      h,b
FCF1    11 F360                 ld      de,seltab       ;set select table address
FCF4    29                      add     hl,hl
FCF5    19                      add     hl,de
FCF6    EB                      ex      de,hl           ;set second address to DE, get seltab to HL
FCF7    09                      add     hl,bc
FCF8    09                      add     hl,bc           ;set first address to HL
FCF9    06 02                   ld      b,2
FCFB    4E              swapl:  ld      c,(hl)          ;swap two bytes
FCFC    1A                      ld      a,(de)
FCFD    77                      ld      (hl),a
FCFE    79                      ld      a,c
FCFF    12                      ld      (de),a
FD00    23                      inc     hl
FD01    13                      inc     de
FD02    10 F7                   djnz    swapl           ;if swap not complete
```

```
FD04    C9                      ret

                        ;;      lcp - load configuration parameters.
                        ;
FD05    3E 81           lcp:    ld      a,10000001b     ;default i/o byte to CRT: and LPT:
FD07    32 0003                 ld      (iobyte),a
FD0A    3A FCD4                 ld      a,(boots)       ;get boot diskette type
FD0D    B7                      or      a
FD0E    0E 20                   ld      c,32
FD10    21 ED80                 ld      hl,bootbf       ;use boot loader buffer
FD13    28 08                   jr      z,lcp1          ;if rigid, use system track, sector 32
FD15    FE 1B                   cp      26+1
FD17    D8                      ret     c               ;no parameters from single density boots
FD18    0E 03                   ld      c,3             ;dd configuration comes from track 0, sector 3
FD1A    21 EE00                 ld      hl,bootbf+128   ;use second half of boot loader buffer
FD1D    CD FA48         lcp1:   call    read
FD20    C0                      ret     nz              ;if can't read configuration
FD21    3A EE00                 ld      a,(bootbf+128)
FD24    D6 E5                   sub     0e5h
FD26    C8                      ret     z
FD27    3A EE7B                 ld      a,(z.xonp)      ;configure Xon-Xoff
FD2A    B7                      or      a
FD2B    28 03                   jr      z,lcp2
FD2D    FE C9                   cp      0c9h
FD2F    C0                      ret     nz
FD30    32 F115         lcp2:   ld      (Xonenb),a
FD33    3A EE60                 ld      a,(z.stpr)      ;configure step rate
FD36    32 FF54                 ld      (steprt),a
FD39    3A EE5F                 ld      a,(z.scra)      ;configure screen attribute
FD3C    32 FD49                 ld      (lcpa),a
FD3F    3A EE62                 ld      a,(z.keym)      ;configure keyboard mask
FD42    32 FD4B                 ld      (lcpb),a
FD45    CD FC3D                 call    pnext
FD48    1B                      defb    esc
FD49    00              lcpa:   defb    0
FD4A    1B                      defb    esc
FD4B    00              lcpb:   defb    0
FD4C    04                      defb    eot
FD4D    21 EE63                 ld      hl,z.sioA       ;configure Sio channels
FD50    3E 02                   ld      a,2
FD52    46              lcp3:   ld      b,(hl)          ;get number of bytes
FD53    23                      inc     hl
FD54    4E                      ld      c,(hl)          ;get port address
FD55    23                      inc     hl
FD56    ED B3                   otir
FD58    3D                      dec     a
FD59    20 F7                   jr      nz,lcp3
FD5B    3A EE7D                 ld      a,(z.baua)      ;configure channel A bit rate
FD5E    D3 00                   out     (bauda),a
FD60    3A EE7E                 ld      a,(z.baub)      ;configure channel B bit rate
FD63    D3 0C                   out     (baudb),a
FD65    3A EE77                 ld      a,(z.siom)      ;configure printer ready mask
FD68    32 F10C                 ld      (siomsk),a
FD6B    3A EE79                 ld      a,(z.siov)      ;configure printer ready value
FD6E    32 F10E                 ld      (sioval),a
FD71    3A EE7F                 ld      a,(z.iobt)      ;configure I/O byte
```

```
FD74    32 0003                     ld      (iobyte),a
FD77    C9                          ret
                                    else
                            boot    equ     what
                                    endif

                            ;;      -- goto to memory location command --
                            ;
                                    overlay goto
0268'                   +           c&seg

FC55    B7                          or      a
FC56    37                          scf
FC57    C8                          ret     z               ;if no parameters
FC58    E5                          push    hl              ;set goto address
FC59    DD E1                       pop     ix              ;ld     ix,hl
FC5B    EB                          ex      de,hl           ;set second arg to HL
FC5C    7D                          ld      a,l             ;and A
FC5D    50                          ld      d,b             ;set third arg to DE
FC5E    5D                          ld      e,l
FC5F    ED 4B FFBB                  ld      bc,(param4)     ;set fourth arg to BC
FC63    CD FAD6                     call    jpix
FC66    CD FC1B                     call    put2hs          ;print A reg
FC69    C3 FC16                     jp      put4hs

                            ;;      -- memory dump command --
                            ;
                                    overlay memdmp
0282'                   +           c&seg

FC55    3D                          dec     a               ;check parameter count
FC56    28 06                       jr      z,mdmp2
FC58    3D                          dec     a
FC59    28 08                       jr      z,mdmp3
FC5B    2A FFBD             mdmp1:  ld      hl,(last)
FC5E    11 0010             mdmp2:  ld      de,16
FC61    18 0E                       jr      mdmp3b

FC63    EB                  mdmp3:  ex      de,hl
FC64    ED 52                       sbc     hl,de           ;derive bytecount for dump range
FC66    D8                          ret     c               ;if addresses backwards
FC67    06 04                       ld      b,4
FC69    CB 3C               mdmp3a: srl     h               ;divide bytecount by 16
FC6B    CB 1D                       rr      l
FC6D    10 FA                       djnz    mdmp3a
FC6F    23                          inc     hl
FC70    EB                          ex      de,hl
FC71    CD FB8F             mdmp3b: call    dump            ;dump de*16 bytes strting at hl
FC74    22 FFBD                     ld      (last),hl
FC77    C9                          ret

                            ;       -- memory examine command --
                            ;
                                    overlay view
02A5'                   +           c&seg

```

```
        FC55    CD FB22         view0:  call    mdata
        FC58    CD FC27                 call    echo
        FC5B    FE 0D                   cp      cr
        FC5D    28 2F                   jr      z,view4
        FC5F    FE 2D                   cp      '-'
        FC61    28 2D                   jr      z,view5
        FC63    FE 22                   cp      '"'
        FC65    20 08                   jr      nz,view1
        FC67    CD F009                 call    conin
        FC6A    CD F00C                 call    conout
        FC6D    18 1B                   jr      view3
        FC6F    CD FBF3         view1:  call    hexbin
        FC72    3F                      ccf
        FC73    D0                      ret     nc
        FC74    4F                      ld      c,a
        FC75    87                      add     a,a
        FC76    87                      add     a,a
        FC77    87                      add     a,a
        FC78    87                      add     a,a
        FC79    47                      ld      b,a
        FC7A    CD FC27                 call    echo
        FC7D    D6 0D                   sub     cr
        FC7F    28 08                   jr      z,view2
        FC81    C6 0D                   add     a,cr
        FC83    CD FBF3                 call    hexbin
        FC86    3F                      ccf
        FC87    D0                      ret     nc
        FC88    48                      ld      c,b
        FC89    81              view2:  add     a,c
        FC8A    77              view3:  ld      (hl),a
        FC8B    CD FB0E                 call    check
        FC8E    23              view4:  inc     hl
        FC8F    23                      inc     hl
        FC90    2B              view5:  dec     hl
        FC91    18 C2                   jr      view0

                                        if      options and o.baud
                                ;       -- Baud Rate Command --
                                ;
                                ;       * B<rate> [channel]     ;channel may be 0,1 or A/B
                                ;

                                        overlay baud
        02E3'               +           c&seg

        FC55    B7                      or      a
        FC56    37                      scf
        FC57    C8                      ret     z               ;if no parameters
        FC58    3D                      dec     a
        FC59    20 02                   jr      nz,baud1        ;if channel specified
        FC5B    1E 01                   ld      e,1             ;set channel 1 (B)
        FC5D    CB 43           baud1:  bit     0,e             ;check port
        FC5F    0E 00                   ld      c,bauda         ;set communications port
        FC61    28 02                   jr      z,baud2
        FC63    0E 0C                   ld      c,baudb         ;set printer port
        FC65    ED 69           baud2:  out     (c),l           ;set baud rate
        FC67    AF                      xor     a
```

```
FC68     C9                            ret
                                       else
                              baud     equ     what
                                       endif

                                       if      options and o.disk
                              ;;       -- disk sector read/write command --
                              ;
                              ;        * R <unit> <track> <sector> <address>
                              ;        * W <unit> <track> <sector> <address>
                              ;
                                       overlay dskcmd
02F7'                    +             c&seg

FC55     47                            ld      b,a
FC56     3A FF5C                       ld      a,(linbuf)
FC59     D6 57                         sub     'W'
FC5B     20 1E                         jr      nz,dsk1
FC5D     B0                            or      b
FC5E     20 16                         jr      nz,dsk0
FC60     4F                            ld      c,a
FC61     CD F2FE                       call    Fastcrt
FC64     23                            inc     hl
FC65     11 0011                       ld      de,17
FC68     01 0015                       ld      bc,21
FC6B     EB                            ex      de,hl
FC6C     AF                            xor     a
FC6D     32 FC54                       ld      ($-25),a
FC70     CD F2A3                       call    crtldir
FC73     C3 FC36                       jp      crlf

FC76     3A FC54              dsk0:    ld      a,($-34)
FC79     B7                            or      a
FC7A     C0                            ret     nz
FC7B     78                   dsk1:    ld      a,b
FC7C     FE 04                         cp      4               ;check parameter count
FC7E     37                            scf
FC7F     C0                            ret     nz
FC80     21 FFB5                       ld      hl,param1       ;move parameters to disk command
FC83     11 FA10                       ld      de,phydrv
FC86     01 0007                       ld      bc,3*2+1
FC89     ED A0                         ldi
FC8B     23                            inc     hl              ;skip upper unit
FC8C     ED B0                         ldir
FC8E     05                            dec     b               ;set select operation
FC8F     CD FA51                       call    xqphys          ;execute physical select
FC92     7D                            ld      a,l
FC93     B4                            or      h
FC94     28 16                         jr      z,dskerr        ;if select error
FC96     06 00                         ld      b,0             ;preset write command
FC98     3A FF5C                       ld      a,(linbuf)      ;get command
FC9B     FE 57                         cp      'W'
FC9D     28 01                         jr      z,dsk3          ;if write
FC9F     04                            inc     b
FCA0     CD FA51              dsk3:    call    xqphys          ;execute driver
FCA3     2A FFBB                       ld      hl,(param4)
```

```
FCA6    11 0010                 ld      de,16           ;assume 256-byte sector
FCA9    CA FB8F                 jp      z,dump          ;dump disk read buffer if no error

FCAC    CD FC3D         dskerr: call    pnext
FCAF    44 73 6B 20             defm    'Dsk Err'
FCB3    45 72 72
FCB6    04                      defb    eot
FCB7    C9                      ret
                                else    ;(not disk options)
                        dskcmd  equ     what
                                endif

                                if      options and o.inpc
                        ;;      -- read input port command --
                        ;
                        ;       * I <16-bit port address>
                        ;
                                overlay incmd
035A'               +           c&seg

FC55    3D                      dec     a
FC56    37                      scf
FC57    C0                      ret     nz              ;if not one parameter
FC58    4D                      ld      c,l             ;set input port
FC59    44                      ld      b,h
FC5A    CD FC36         in1:    call    crlf
FC5D    79                      ld      a,c             ;display port address
FC5E    CD FC1B                 call    put2hs
FC61    ED 78           in2:    in      a,(c)
FC63    CD FC1B                 call    put2hs
FC66    CD F009                 call    conin           ;read character
FC69    FE 20                   cp      ' '
FC6B    28 F4                   jr      z,in2           ;read same port again
FC6D    FE 0D                   cp      cr
FC6F    28 06                   jr      z,in3           ;if read next
FC71    FE 2D                   cp      '-'
FC73    28 04                   jr      z,in4           ;if read previous
FC75    B7                      or      a               ;clear carry
FC76    C9                      ret
FC77    03              in3:    inc     bc              ;advance to next port
FC78    03                      inc     bc
FC79    0B              in4:    dec     bc
FC7A    18 DE                   jr      in1
                                else
                        incmd   equ     what
                                endif

                                if      options and o.outc
                        ;;      -- write to output port command --
                        ;
                        ;       * O <16-bit port address> <8-bit value>
                        ;
                                overlay outcmd
0381'               +           c&seg

FC55    FE 02                   cp      2               ;require two parameters
```

```
FC57    37                      scf
FC58    C0                      ret     nz              ;if not 2 parameters
FC59    4D                      ld      c,l             ;set 16 bit output port address
FC5A    44                      ld      b,h
FC5B    ED 59                   out     (c),e           ;output to d0-d7 and address to a0-a17
FC5D    B7                      or      a
FC5E    C9                      ret
                                else
                        outcmd  equ     what
                                endif

                                if      options and o.ramt
                        ;;      -- memory read/write diagnostic command --
                        ;
                        ;       * X <first addr> <last addr>
                        ;
                                overlay test
038B'               +           c&seg

FC55    FE 02                   cp      2               ;check parameter count
FC57    37                      scf
FC58    C0                      ret     nz
FC59    13                      inc     de
FC5A    5A                      ld      e,d             ;get ending page address into e
FC5B    54                      ld      d,h             ;get starting page address into d
FC5C    06 00                   ld      b,0             ;initialize pass counter
FC5E    62              test1:  ld      h,d             ;point hl to start of block
FC5F    2E 00                   ld      l,0
FC61    7D              test2:  ld      a,l
FC62    AC                      xor     h               ;generate test byte
FC63    A8                      xor     b
FC64    77                      ld      (hl),a          ;store byte in ram
FC65    23                      inc     hl
FC66    7C                      ld      a,h
FC67    BB                      cp      e               ;check for end of test block
FC68    20 F7                   jr      nz,test2
FC6A    62                      ld      h,d             ;now read back each byte & compare
FC6B    2E 00                   ld      l,0             ;point hl back to start
FC6D    7D              test3:  ld      a,l
FC6E    AC                      xor     h               ;re-generate test byte data
FC6F    A8                      xor     b
FC70    CD FB0E                 call    check           ;verify memory data still good
FC73    C0                      ret     nz              ;exit if escape request is indicated
FC74    23                      inc     hl              ; else go on to next byte
FC75    7C                      ld      a,h
FC76    BB                      cp      e               ;check for end of block
FC77    20 F4                   jr      nz,test3
FC79    04                      inc     b               ;bump pass count
FC7A    3E 2B                   ld      a,'+'
FC7C    CD FC20                 call    output
FC7F    28 DD                   jr      z,test1         ;do another pass if user not unhappy
FC81    C9                      ret
                                else
                        test    equ     what
                                endif
```

```
                                    if      options and o.fill
                            ;;      -- fill memory with constant command --
                            ;
                                    overlay fill
03B8'                  +            c&seg

FC55    FE 03                       cp      3               ;check if parameter count=3
FC57    37                          scf
FC58    C0                          ret     nz
FC59    71                  filll:  ld      (hl),c
FC5A    E5                          push    hl
FC5B    B7                          or      a
FC5C    ED 52                       sbc     hl,de           ;compare hl to end address in de
FC5E    E1                          pop     hl
FC5F    23                          inc     hl              ;advance pointer after comparison
FC60    38 F7                       jr      c,filll
FC62    C9                          ret
                                    else
                            fill    equ     what
                                    endif

                                    if      options and o.move
                            ;;      -- memory block move command --
                            ;
                                    overlay block
03C6'                  +            c&seg

FC55    FE 03                       cp      3               ;check if parameter count=3
FC57    37                          scf
FC58    C0                          ret     nz
FC59    CD FC49                     call    blocad
FC5C    79                          ld      a,c
FC5D    B0                          or      b
FC5E    C8                          ret     z               ;exit now if bc=0
FC5F    ED B0                       ldir
FC61    C9                          ret
                                    else
                            block   equ     what
                                    endif

                                    if      options and o.verf
                            ;;      -- memory block compare command --
                            ;
                                    overlay vercmd
03D3'                  +            c&seg

FC55    FE 03                       cp      3               ;check if parameter count=3
FC57    37                          scf
FC58    C0                          ret     nz
FC59    CD FC49                     call    blocad
FC5C    18 08                       jr      verf2

FC5E    1A                  verf1:  ld      a,(de)
FC5F    CD FB0E                     call    check           ;compare data @ de and @ hl
FC62    C0                          ret     nz              ;exit if escape request is indicated
FC63    23                          inc     hl
```

```
FC64   13                    inc     de
FC65   0B                    dec     bc
FC66   78            verf2:  ld      a,b
FC67   B1                    or      c
FC68   20 F4                 jr      nz,verf1
FC6A   C9                    ret
                             else
                     vercmd  equ     what
                             endif

                             if      options and o.prot
                     ;;      Printer Protocol.
                     ;
                             overlay proto
03E9'           +            c&seg

FC55   3D                    dec     a
FC56   28 10                 jr      z,prot1         ;if one parameter
FC58   D6 02                 sub     2
FC5A   37                    scf
FC5B   C0                    ret     nz
FC5C   7B                    ld      a,e
FC5D   F6 04                 or      4
FC5F   32 F10C               ld      (siomsk),a
FC62   79                    ld      a,c
FC63   F6 04                 or      4
FC65   32 F10E               ld      (sioval),a
FC68   7D            prot1:  ld      a,l
FC69   B7                    or      a
FC6A   3E C9                 ld      a,0c9h
FC6C   28 01                 jr      z,prot2
FC6E   AF                    xor     a
FC6F   32 F115       prot2:  ld      (Xonenb),a
FC72   C9                    ret
                             else
                     proto   equ     what
                             endif

                             if      (options and o.type) ne 0
                     ;;      Type - Simple Typewriter.
                     ;
                             overlay type
0407'           +            c&seg

FC55   B7                    or      a
FC56   28 06                 jr      z,typ0          ;if no baud rate
FC58   3D                    dec     a
FC59   37                    scf
FC5A   C0                    ret     nz              ;if more than one parameter
FC5B   7D                    ld      a,l
FC5C   D3 0C                 out     (baudb),a       ;set printer baud rate
FC5E   CD FC3D       typ0:   call    pnext
FC61   1A                    defb    clrs
                             if      (options and o.esct) ne 0
FC62   1B 31                 defb    esc,'1'         ;set 8 bit keyboard mode
                             else
```

```
                                          defb    '0'-'@'         ;set 8 bit keyboard mode
                                          endif
        FC64    54 79 70 65               defm    'Typewriter mode.  Touch CTRL+ESC to exit.'
        FC68    77 72 69 74
        FC6C    65 72 20 6D
        FC70    6F 64 65 2E
        FC74    20 20 54 6F
        FC78    75 63 68 20
        FC7C    43 54 52 4C
        FC80    2B 45 53 43
        FC84    20 74 6F 20
        FC88    65 78 69 74
        FC8C    2E
        FC8D    0D 0A 04                  defb    cr,lf,eot

        FC90    CD F0E5           typ1:   call    siost           ;status printer
        FC93    28 0F                     jr      z,typ2          ;if char not ready
        FC95    CD F0F0                   call    sioin           ;read character
        FC98    E6 7F                     and     7fh             ;strip parity bit
        FC9A    CD F00C                   call    conout
        FC9D    FE 0D                     cp      cr
        FC9F    3E 0A                     ld      a,lf
        FCA1    CC F00C                   call    z,conout
        FCA4    CD F006           typ2:   call    const           ;status console
        FCA7    28 E7                     jr      z,typ1          ;if user not active
        FCA9    CD F009                   call    conin           ;read keyboard
        FCAC    CD F0F8                   call    sioout          ;send character to printer
        FCAF    CD F00C                   call    conout          ;and screen
        FCB2    FE 0D                     cp      cr
        FCB4    20 DA                     jr      nz,typ1         ;if not CR
        FCB6    3E 0A                     ld      a,lf            ;send line feed to screen and printer
        FCB8    CD F0F8                   call    sioout
        FCBB    CD F00C                   call    conout
        FCBE    18 D0                     jr      typ1
                                          else
                                  type    equ     what
                                          endif

                                          if      options and o.term
                                  ;;      Terminal / Scroll Driver.
                                  ;
                                          subttl  Terminal / Screen Manager
                                          page
```

```


                                        overlay term
0472'                      +            c&seg

000F                            pass8   equ     15
0016                            inslin  equ     22
0017                            dellin  equ     23
001A                            clrchr  equ     26
001E                            homscr  equ     30
001F                            force   equ     31

0081                            kuplin  equ     81h             ;Move top line off screen to buffer
0082                            kdnlin  equ     82h             ;Move bottom line off screen to buffer
00B1                            Rmttog  equ     80h+'1'         ;Toggle Remote Echo
00B2                            Rmtalf  equ     80h+'2'         ;Toggle Remote Auto LF after CR
008A                            Localf  equ     80h+lf          ;Toggle Local Auto LF after CR
00FF                            Typtog  equ     80h+7fh         ;Toggle Local Echo
00AE                            Brkkey  equ     80h+'.'         ;Hardware BREAK function

0007                            s.lecho equ     7               ;local echo
0006                            s.recho equ     6               ;remote echo
0005                            s.autol equ     5               ;local auto lf after cr
0004                            s.autor equ     4               ;remote auto lf after cr

0100                            Trmbuf  equ     100h
EE80                            Buftop  equ     Trmbuf+760*80
EF00                            Siobuf  equ     Monitr-100h
EF00                            Trmstk  equ     Siobuf

FC55    FE 02                           cp      2               ;check number of arguments
FC57    3F                              ccf
FC58    D8                              ret     c               ;if more than 1
FC59    B7                              or      a
FC5A    20 02                           jr      nz,term1        ;if port specifed
FC5C    2E 00                           ld      l,0
FC5E    01 0406                 term1:  ld      bc,siocpa+siodpa*256    ;preset A channel ports
FC61    CB 45                           bit     0,l
FC63    28 03                           jr      z,term2         ;if 0/1 or A/B
FC65    01 0507                         ld      bc,siocpb+siodpb*256    ;set B channel ports
FC68    ED 43 FE78              term2:  ld      (ports),bc
FC6C    31 EF00                         ld      sp,trmstk
FC6F    CD FC3D                         call    pnext
FC72    1A                              db      clrs
                                        if      options and o.esct
FC73    1B 31                           db      esc,'1'
                                        else
                                        db      pass8
                                        endif
FC75    54 65 72 6D                     db      'Terminal mode.  Touch CTRL+ESC to exit.'
FC79    69 6E 61 6C
FC7D    20 6D 6F 64
FC81    65 2E 20 20
FC85    54 6F 75 63
FC89    68 20 43 54
```

```
FC8D    52 4C 2B 45
FC91    53 43 20 74
FC95    6F 20 65 78
FC99    69 74 2E
FC9C    0D 0A                         db      cr,lf
FC9E    04                            db      eot

FC9F    FD 21 FEE5                    ld      iy,status          ;set pointer to status byte

FCA3    CD F006             term3:    call    const              ;status keyboard
FCA6    C4 FCB1                       call    nz,pki             ;process keyboard input
FCA9    CD FEC8                       call    sioist             ;status sio
FCAC    C4 FD40                       call    nz,prc             ;process remote character
FCAF    18 F2                         jr      term3              ;until user escapes

                            ;;        pki - Process Keyboard Input.
                            ;
FCB1    3E 00               pki:      ld      a,0
FCB2                        brkflg    equ     $-1
FCB3    B7                            or      a
FCB4    C4 FD21                       call    nz,clrbrk          ;terminate pending break
FCB7    CD F009                       call    conin              ;read input
FCBA    CB 7F                         bit     7,a
FCBC    20 0B                         jr      nz,pki1            ;if function key
FCBE    CD FD5F                       call    sndrmt             ;send it to remote
FCC1    FD CB 00 7E                   bit     s.lecho,(iy)
FCC5    C4 FD4C                       call    nz,sndloc          ;if local echo enabled, display console input
FCC8    C9                            ret
FCC9    FE 81               pki1:     cp      kuplin
FCCB    20 12                         jr      nz,pki2            ;if not scroll up
FCCD    CD FE50                       call    gcp                ;get cursor position
FCD0    CD FC3D                       call    pnext              ;home screen, then delete top line
                                      if      options and o.esct
FCD3    1E 1B 52 04                   defb    homscr,esc,'R',eot
                                      else
                                      defb    homscr,dellin,eot
                                      endif
FCD7    CD FD89                       call    ltl                ;link top line
FCDA    CD FDEB                       call    dbl                ;display bottom line
FCDD    18 14                         jr      pki3
FCDF    FE 82               pki2:     cp      kdnlin
FCE1    20 13                         jr      nz,pki4            ;if not scroll down
FCE3    CD FE50                       call    gcp                ;get cursor position
FCE6    CD FC3D                       call    pnext              ;home screen, insert blank line
                                      if      options and o.esct
FCE9    1E 1B 45 04                   defb    homscr,esc,'E',eot
                                      else
                                      defb    homscr,dellin,eot
                                      endif
FCED    CD FE0B                       call    lbl                ;link bottom line
FCF0    CD FDB1                       call    dtl                ;display top line
FCF3    C3 FE6E             pki3:     jp      rcp                ;restore cursor position
FCF6    FE FF               pki4:     cp      Typtog
FCF8    20 04                         jr      nz,pki5
FCFA    3E 80                         ld      a,1 shl s.lecho
FCFC    18 16                         jr      pki8
```

```
FCFE    FE B1           pki5:   cp      Rmttog
FD00    20 04                   jr      nz,pki6
FD02    3E 40                   ld      a,1 shl s.recho
FD04    18 0E                   jr      pki8
FD06    FE 8A           pki6:   cp      Localf
FD08    20 04                   jr      nz,pki7
FD0A    3E 20                   ld      a,1 shl s.autol
FD0C    18 06                   jr      pki8
FD0E    FE B2           pki7:   cp      Rmtalf
FD10    20 09                   jr      nz,pki9
FD12    3E 10                   ld      a,1 shl s.autor
FD14    FD AE 00        pki8:   xor     (iy)
FD17    FD 77 00                ld      (iy),a
FD1A    C9                      ret
FD1B    FE AE           pki9:   cp      Brkkey
FD1D    C0                      ret     nz
FD1E    3A FCB2                 ld      a,(brkflg)
FD21    EE FF           clrbrk: xor     -1
FD23    32 FCB2                 ld      (brkflg),a
FD26    16 10                   ld      d,10h           ;set line SPACING
FD28    20 02                   jr      nz,setbrk
FD2A    16 00                   ld      d,0             ;set line MARKING
FD2C    ED 4B FE78      setbrk: ld      bc,(ports)
FD30    3E 05                   ld      a,5             ;set up WR5
FD32    F3                      di
FD33    ED 79                   out     (c),a
FD35    3E AA                   ld      a,10101010b     ;assert DTR, 7 bpc, RTS, Tx Enb
FD37    B2                      or      d
FD38    ED 79                   out     (c),a
FD3A    FB                      ei
FD3B    3E FF                   ld      a,0ffh
FD3D    C3 FE90                 jp      sioot           ;send RUBOUT to allow MARKING

                        ;;      prc - Process Remote Character.
                        ;
FD40    CD FED6         prc:    call    sioinc          ;read remote character
FD43    FD CB 00 76             bit     s.recho,(iy)
FD47    C4 FD5F                 call    nz,sndrmt       ;echo it back
FD4A    18 26                   jr      doc             ;display it locally

                        ;;      sndloc - send character to screen.
                        ;
FD4C    CD FD72         sndloc: call    doc
FD4F    FE 0D                   cp      cr
FD51    C0                      ret     nz
FD52    FD CB 00 6E             bit     s.autol,(iy)
FD56    C8                      ret     z
FD57    3E 0A                   ld      a,lf
FD59    CD FD72                 call    doc
FD5C    3E 0D                   ld      a,cr
FD5E    C9                      ret

                        ;;      sndrmt - send character to remote.
                        ;
FD5F    CD FE90         sndrmt: call    sioot
FD62    FE 0D                   cp      cr
```

```
FD64     C0                            ret     nz
FD65     FD CB 00 66                   bit     s.autor,(iy)
FD69     C8                            ret     z
FD6A     3E 0A                         ld      a,lf
FD6C     CD FE90                       call    sioot
FD6F     3E 0D                         ld      a,cr
FD71     C9                            ret

                               ;;      doc - Display One Character.
                               ;
FD72     FE 7F                 doc:    cp      7fh
FD74     C8                            ret     z               ;don't display RUBOUT
FD75     4F                            ld      c,a             ;send it to screen
FD76     C5                            push    bc
FD77     CD FE9B                       call    outcrt          ;display character
FD7A     C1                            pop     bc
FD7B     47                            ld      b,a
FD7C     79                            ld      a,c
FD7D     FE 0A                         cp      lf
FD7F     C0                            ret     nz
FD80     04                            inc     b
FD81     05                            dec     b
FD82     C0                            ret     nz              ;if line feed did not scroll
FD83     CD FD89                       call    ltl             ;link top line
FD86     3E 0A                         ld      a,lf
FD88     C9                            ret

                               ;;      ltl - link top line.
                               ;
FD89     21 FF5C               ltl:    ld      hl,linbuf
FD8C     ED 5B FEE8                    ld      de,(topptr)     ;set address of line above screen
FD90     01 0050                       ld      bc,80
FD93     ED B0                         ldir                    ;move line
FD95     CD FE34                       call    wup             ;wrap upper pointer
FD98     ED 53 FEE8                    ld      (topptr),de     ;set new top line address
FD9C     2A FEE6                       ld      hl,(botptr)
FD9F     EB                            ex      de,hl
FDA0     B7                            or      a
FDA1     ED 52                         sbc     hl,de
FDA3     C0                            ret     nz
FDA4     11 0050                       ld      de,80
FDA7     19                            add     hl,de           ;advance bottom pointer
FDA8     EB                            ex      de,hl
FDA9     CD FE34                       call    wup             ;wrap upper pointer
FDAC     ED 53 FEE6                    ld      (botptr),de
FDB0     C9                            ret

                               ;;      dtl - Display Top Line.
                               ;
FDB1     ED 5B FEE8            dtl:    ld      de,(topptr)     ;get line above screen
FDB5     21 FFB0                       ld      hl,-80
FDB8     19                            add     hl,de
FDB9     CD FE41                       call    wlp             ;wrap lower pointer
FDBC     22 FEE8                       ld      (topptr),hl
FDBF     01 0050                       ld      bc,80
```

```
                                  ;;      dln - display line.
                                  ;
FDC2    E5                        dln:    push    hl
FDC3    09                                add     hl,bc
FDC4    2B                                dec     hl
FDC5    3E 20                             ld      a,' '
FDC7    ED A9                     dln1:   cpd
FDC9    20 03                             jr      nz,dln2         ;if not trailing blank
FDCB    EA FDC7                           jp      pe,dln1
FDCE    E1                        dln2:   pop     hl
FDCF    E0                                ret     po              ;if entire line blank
FDD0    41                                ld      b,c
FDD1    04                                inc     b
FDD2    C5                        dln3:   push    bc
FDD3    7E                                ld      a,(hl)
FDD4    4F                                ld      c,a
FDD5    FE 20                             cp      ' '
FDD7    30 08                             jr      nc,dln4
FDD9    E5                                push    hl
FDDA    0E 1F                             ld      c,force         ;force next character out
FDDC    CD FE9B                           call    outcrt
FDDF    E1                                pop     hl
FDE0    4E                                ld      c,(hl)
FDE1    23                        dln4:   inc     hl              ;advance address
FDE2    E5                                push    hl
FDE3    CD FE9B                           call    outcrt          ;display character
FDE6    E1                                pop     hl
FDE7    C1                                pop     bc
FDE8    10 E8                             djnz    dln3            ;if not entire line
FDEA    C9                                ret

                                  ;;      dbl - Display bottom line.
                                  ;
FDEB    CD FC3D                   dbl:    call    pnext           ;plant cursor on bottom line
FDEE    1B 3D 37 20                       db      esc,'=',' '+23,' ',eot
FDF2    04
FDF3    2A FEE6                           ld      hl,(botptr)
FDF6    E5                                push    hl
FDF7    01 004F                           ld      bc,80-1
FDFA    CD FDC2                           call    dln             ;display bottom line
FDFD    E1                                pop     hl
FDFE    01 0050                           ld      bc,80
FE01    09                                add     hl,bc
FE02    EB                                ex      de,hl
FE03    CD FE34                           call    wup
FE06    ED 53 FEE6                        ld      (botptr),de
FE0A    C9                                ret

                                  ;;      lbl - link bottom line.
                                  ;
FE0B    01 0050                   lbl:    ld      bc,80
FE0E    2A FEE6                           ld      hl,(botptr)
FE11    B7                                or      a
FE12    ED 42                             sbc     hl,bc
FE14    CD FE41                           call    wlp
FE17    22 FEE6                           ld      (botptr),hl
```

```
FE1A   EB                   ex      de,hl
FE1B   2A FEE8              ld      hl,(topptr)
FE1E   B7                   or      a
FE1F   ED 52                sbc     hl,de
FE21   20 0B                jr      nz,lbl1
FE23   2A FEE8              ld      hl,(topptr)
FE26   ED 42                sbc     hl,bc
FE28   CD FE41              call    wlp
FE2B   22 FEE8              ld      (topptr),hl
FE2E   21 FF5C      lbl1:   ld      hl,linbuf
FE31   ED B0                ldir                    ;move gold mine to buffer
FE33   C9                   ret

                    ;;      wup - Wrap upper pointer.
                    ;
FE34   E5           wup:    push    hl
FE35   21 EE80              ld      hl,buftop       ;set end of buffer address
FE38   37                   scf
FE39   ED 52                sbc     hl,de
FE3B   E1                   pop     hl
FE3C   D0                   ret     nc              ;if not past end of buffer
FE3D   11 0100              ld      de,Trmbuf       ;start over at beggining of buffer
FE40   C9                   ret

                    ;;      wlp - Wrap lower pointer.
                    ;
FE41   E5           wlp:    push    hl
FE42   D5                   push    de
FE43   11 0100              ld      de,Trmbuf       ;set start of buffer address
FE46   B7                   or      a
FE47   ED 52                sbc     hl,de
FE49   D1                   pop     de
FE4A   E1                   pop     hl
FE4B   D0                   ret     nc              ;if not below start of buffer
FE4C   21 EE30              ld      hl,buftop-80    ;start over at end of buffer
FE4F   C9                   ret

                    ;;      gcp - get cursor position.
                    ;
FE50   ED 4B FFB1   gcp:    ld      bc,(base)
FE54   2A FFAC              ld      hl,(cursor)
FE57   45                   ld      b,l             ;get column to B
FE58   CB B8                res     7,b
FE5A   29                   add     hl,hl           ;set screen row to H
FE5B   7C                   ld      a,h
FE5C   D6 60                sub     crtbas*2
FE5E   91                   sub     c               ;row = 23-(base-curh)
FE5F   C6 17                add     a,23
FE61   D6 18        gcp1:   sub     24
FE63   30 FC                jr      nc,gcp1
FE65   4F                   ld      c,a             ;set row
FE66   21 1F38              ld      hl,'  '+24-100h ;offset for <esc>=
FE69   09                   add     hl,bc
FE6A   22 FE73              ld      (rcpa),hl
FE6D   C9                   ret
```

```
                                ;;      rcp - Restore cursor position.
                                ;
        FE6E    CD FC3D         rcp:    call    pnext
        FE71    1B 3D                   db      esc,'='
        FE73    20 20           rcpa:   db      '  '
        FE75    04                      db      eot
        FE76    C9                      ret

                                ;;      sio drivers.
                                ;
        FE77    01 FE78         sioins: ld      bc,ports        ;set status port to c
        FE78                    ports   equ     $-2
        FE7A    ED 78                   in      a,(c)
        FE7C    CB 47                   bit     0,a             ;test rca
        FE7E    C9                      ret

        FE7F    CD FE77         sioinp: call    sioins          ;get status
        FE82    28 FB                   jr      z,sioinp        ;if not ready
        FE84    48                      ld      c,b             ;set data port address
        FE85    ED 78                   in      a,(c)
        FE87    CB BF                   res     7,a             ;pitch parity bit
        FE89    C9                      ret

        FE8A    CD FE77         siordt: call    sioins          ;get sio status
        FE8D    CB 57                   bit     2,a             ;test TX empty
        FE8F    C9                      ret

        FE90    08              sioot:  ex      af,af'          ;save char to send
        FE91    CD FE8A         siootl: call    siordt          ;test transmit ready status
        FE94    28 FB                   jr      z,siootl        ;if not ready
        FE96    48                      ld      c,b
        FE97    08                      ex      af,af'
        FE98    ED 79                   out     (c),a
        FE9A    C9                      ret

        FE9B    CD FEA7         outcrt: call    siopl           ;poll for input before & after
        FE9E    CD F2FE                 call    fastcrt
        FEA1    F5                      push    af              ;save balcones gold
        FEA2    CD FEA7                 call    siopl
        FEA5    F1                      pop     af
        FEA6    C9                      ret

                                ;;      Siopl - Sio Poll Input Characters.
                                ;
        FEA7    C5              Siopl:  push    bc
        FEA8    CD FE77                 call    sioins          ;input Sio status
        FEAB    28 19                   jr      z,siopl3        ;if input not ready
        FEAD    48                      ld      c,b             ;set data port address
        FEAE    ED 78                   in      a,(c)
        FEB0    CB BF                   res     7,a             ;pitch parity bit
        FEB2    E5                      push    hl
        FEB3    2A FEEA                 ld      hl,(ipoint)     ;set in pointer
        FEB6    77                      ld      (hl),a          ;store character in fifo
        FEB7    2C                      inc     l
        FEB8    20 02                   jr      nz,siopl1
        FEBA    2E 00                   ld      l,low siobuf
```

```
FEBC    3A FEEC             siopl1: ld      a,(opoint)
FEBF    95                          sub     l
FEC0    28 03                       jr      z,siopl2        ;if buffer full
FEC2    22 FEEA                     ld      (ipoint),hl
FEC5    E1                  siopl2: pop     hl
FEC6    C1                  siopl3: pop     bc
FEC7    C9                          ret

                            ;;      Sioist - Sio Input Status.
                            ;
FEC8    CD FEA7             Sioist: call    Siopl           ;poll for input
FECB    2A FEEC                     ld      hl,(opoint)     ;set out pointer
FECE    3A FEEA                     ld      a,(ipoint)
FED1    95                          sub     l
FED2    C8                          ret     z               ;if data not ready
FED3    F6 FF                       or      -1
FED5    C9                          ret

                            ;;      Sioin - Sio Input Character.
                            ;
FED6    CD FEC8             Sioinc: call    Sioist          ;set input ready status
FED9    28 FB                       jr      z,Sioinc
FEDB    7E                          ld      a,(hl)
FEDC    2C                          inc     l               ;advance out
FEDD    20 02                       jr      nz,Sioil
FEDF    2E 00                       ld      l,low siobuf
FEE1    22 FEEC             Sioil:  ld      (opoint),hl
FEE4    C9                          ret

FEE5    00                  status: db      0

FEE6    0100                botptr: dw      Trmbuf
FEE8    0100                topptr: dw      Trmbuf

FEEA    EF00                ipoint: dw      siobuf
FEEC    EF00                opoint: dw      siobuf
                                    else
                            term    equ     what
                                    endif

                                    if      options and o.help

                            ;;      Help Key Command.
                            ;
                                    overlay help
070B'               +               c&seg

FC55    CD FC3D                     call    pnext
FC58    42 61 75 64                 defb    'Baud           <rate> [B/A]',cr,lf
FC5C    09 09 3C 72
FC60    61 74 65 3E
FC64    20 5B 42 2F
FC68    41 5D 0D 0A
FC6C    44 75 6D 70                 defb    'Dump           [start] [end]',cr,lf
FC70    09 09 5B 73
FC74    74 61 72 74
```

```
        FC78    5D 20 5B 65
        FC7C    6E 64 5D 0D
        FC80    0A
        FC81    47 6F 74 6F                     defb    'Goto           <addr>',cr,lf
        FC85    09 09 3C 61
        FC89    64 64 72 3E
        FC8D    0D 0A
        FC8F    4D 6F 64 69                     defb    'Modify         <addr>',cr,lf
        FC93    66 79 09 09
        FC97    3C 61 64 64
        FC9B    72 3E 0D 0A
        FC9F    50 72 6F 74                     defb    'Protocol       <xon> [msk val]',cr,lf
        FCA3    6F 63 6F 6C
        FCA7    09 3C 78 6F
        FCAB    6E 3E 20 5B
        FCAF    6D 73 6B 20
        FCB3    76 61 6C 5D
        FCB7    0D 0A
        FCB9    04                              defb    eot
        FCBA    C9                              ret
                                                else
                                        help    equ     what
                                                endif

                                                subttl  Segment Size Information
                                                page
```

```

                                   ;;      Top of Overlay Area.
                                   ;
                                           overlay stop
      0771'                   +            c&seg
      0299                         tpamax  equ     tpal            ;set length of transient move

                                   ;;      Top of Resident Monitor.
                                   ;
                                           below
      0000!                   +            defs    comres
      041B                         rbase   equ     $

                                   ;;      Top of Non Resident Monitor.
                                   ;
                                           above
      0518!                   +            d&seg
      FC55                         restop  equ     $               ;resident top
      0C55                         reslen  equ     $-monitr        ;length of resident monitor

                                           update                  ;clear active segment

                                   ;;      Top of Burned Rom Set.
                                   ;
      17E1                         romtop  equ     bloc+dloc+tloc-monitr

                                   ;;      Fill Out Unused Rom Space.
                                   ;
      0C55"                                cseg

                                           if      (rom+romsiz-romtop) gt 0
      0771'                                defs    (rom+romsiz-romtop),-1
                                           endif

                                           subttl  Resident Monitor System Ram
                                           page
```

```

                        .phase  ram

                ;;      Start of Documented Storage Locations.
                ;
FF00            vectab:                         ;interrupt vector table starts here
FF00            siovec: defs    16              ;space for 8 vectors for sio
FF10            ctcvec: defs    8               ;space for 4 vectors for ctc
FF18            sysvec: defs    4               ;space for 2 vectors for system pio
FF1C            genvec: defs    4               ;space for 2 vectors for general pio

                ;;      keyboard data input fifo variables
                ;
FF20            fifo:   defs    16              ;console input fifo
FF30            fifcnt: defs    1               ;fifo data counter
FF31            fifin:  defs    1               ;fifi input pointer
FF32            fifout: defs    1               ;fifo output pointer

FF33                    defs    1               ;round address

                ;;      More interrupt vectors
                ;
FF34            expvec: defs    8               ;space for 4 vectors for expansion slot

                ;;      Available memory pointers.
                ;
FF3C            availb: defs    2               ;bottom of available memory
FF3E            availt: defs    2               ;top of available memory

                ;
                ;;      End of documented storage locations.

FF40                    defs    16              ;local stack for interrupts
FF50            intstk:

                ;;      clock-timer interrupt variables
                ;
FF50            Milsec: defs    2               ;One   Millisecond timer, Enable int on ctc1
FF52            tikcnt: defs    2               ;16 bit seconds counter (18 hr, 12 min, 16 sec)
FF54            steprt: defs    1               ;WD 1797 step rate
FF55            timout: defs    1               ;time-out, decrements once per second
                ;
                ;       Getime entry returns the address of DAY
                ;
FF56            day:    defs    1               ;calendar day           (01-31)
FF57            month:  defs    1               ;         month         (01-12)
FF58            year:   defs    1               ;         year-1970     (1970-2225)
FF59            hrs:    defs    1               ;clock hours            (00-23)
FF5A            mins:   defs    1               ;      minutes          (00-59)
FF5B            secs:   defs    1               ;      seconds          (00-59)

                ;;      crt output driver variables
                ;
FF5C            linbuf: defs    80              ;line buffer & Bcc gold mine
FFAC            cursor: defs    2               ;cursor pointer
```

```
FFAE                csrchr: defs    1               ;character used for a cursor
FFAF                dircur: defs    2               ;cursor pointer for direct crt display
FFB1                base:   defs    1               ;current contents of scroll register
FFB2                leadin: defs    1               ;state of lead-in sequence handler
FFB3                attrib: defs    1               ;attribute enable
FFB4                chrsav: defs    1               ;character under cusror

                    ;;      console monitor program variables
                    ;
FFB5                param1: defs    2               ;storage for numbers read
FFB7                param2: defs    2               ; from line input buffer
FFB9                param3: defs    2               ; by 'PARAMS' subroutine
FFBB                param4: defs    2               ; for command processors
FFBD                last:   defs    2               ;last address used by 'MEMDMP'

                    ;;      Configurable parameter address table
                    ;
FFBF                contbl: defs    2*numcon

FFCB                spare1: defs    1               ;spare configuration byte
FFCC                spare2: defs    1               ;another spare byte

FFCD                spare:  defs    (ram+100h-48)-$ ;spare ram space

FFD0                sparnd:                         ;end of spare ram

FFD0                        defs    16              ;crt stack
FFE0                crtstk:

FFE0                rstsp:  defs    2               ;sp register on reset
FFE2                rsthl:  defs    2               ;hl register on reset
FFE4                rstpc:  defs    2               ;possible pc from top of stack

FFE6                        defs    26              ;monitor stack
0000                stack:

                            .dephase

                            subttl  Console Messages
                            page
```

```

::      Console Message Macros.
:
message macro   text,h1,h2
        if1
        .radix  16
        printx  <text>,%(h1),%(h2-1),%((h2)-(h1))
        .radix  10
        endif
        endm

printx  macro   text,h1,h2,h3
        .printx + text h1 - h2 = h3 +
        endm

        if      romtop ge (rom+romsiz)
        message <* The ROM set is Too big *>,rom+romsiz,romtop
        endif

        if      cloc+tpal gt ram
        message <* The TPA set is Too big *>,ram,cloc+tpal
        endif

        message <Non-resident executes >,rom,bloc
        message <Rom is burned up from >,rom,romtop
        message <Unused Rom Space from >,romtop,rom+romsiz-1
        message <Resident Monitor needs>,monitr,rqtop
        message <Space Wasted to Driver>,Wasted,Seltab
        message <Physical Disk Drivers >,Seltab,Dvrlmt
        message <Driver Offset for ZSID>,200h+bloc+Seltab-Monitr,0
        message <I/O Byte Drivers from >,iobloc,iobloc+iobdvs
        message <Command Processor Area>,rqtop,restop
        message <Transient Overlay ROM >,start,stop
        message <Transient Command Area>,cloc,cloc+tpal
        message <Spare Locations in Ram>,spare,sparnd

        subttl  The*End
        end     entry
```

```
Balcones Operating System for the XEROX 820-II  MACRO-80 3.44   09-Dec-81
The*End

Macros:
ABOVE           BELOW           BSEG            MESSAGE         OVERLAY
PHEX            PMSG            PRINTX          SEGMENT         SERVICE
UPDATE

Symbols:
0080    ..              F360    .A              F362    .B
F364    .C              F366    .D              F368    .E
F36A    .F              F36C    .G              F36E    .H
F370    .I              F372    .J              F374    .K
F376    .L              F378    .M              F37A    .N
F37C    .O              F37E    .P              009B    ABORT
F6F2    ADDRH           F6F3    ADDRL           FD80    ALLO0
FDA0    ALLO1           FDC0    ALLO2           FDE0    ALLO3
FE00    ALLO4           FE80    ALLO5           FEC0    ALLO6
FEE0    ALLO7           0068    ASYNC           FFB3    ATTRIB
FA78    AUTOBT          FF3C    AVAILB          FF3E    AVAILT
0000    B.BSY           0002    B.CD            0004    B.IO
0001    B.MSG           0006    B.PAR           0003    B.REQ
0007    B.RST           0005    B.SEL           02CE    BAKSPC
FFB1    BASE            1353    BAUD            FC5D    BAUD1
FC65    BAUD2           0000    BAUDA           000C    BAUDB
0018!   BBASE           040F    BBG             032F    BELL
0331    BELL1           0028    BELLOF          0029    BELLON
041B    BLOC            FC49    BLOCAD          1436    BLOCK
0061    BLOFC           0061    BLONC           0035    BLTIM
000F    BNDRY           11B8    BOOT            FC58    BOOT1
FC69    BOOT2           FCA3    BOOT3           ED80    BOOTBF
FCDD    BOOTD           FCD9    BOOTER          0080    BOOTLD
FCD4    BOOTS           FEE6    BOTPTR          FCB2    BRKFLG
00AE    BRKKEY          0000    BSPACE          EE80    BUFTOP
0004    C.8IN           0004    C.FIVE          0007    C.FLAW
00C0    C.FLPY          0004    C.FMAT          0006    C.FTRK
000C    C.INIT          0007    C.KEYM          0008    C.READ
0001    C.RECAL         0003    C.RQSN          0002    C.RSYN
0006    C.SASI          000B    C.SEEK          00E0    C.TRAM
0000    C.TRDY          0005    C.TWO           0005    C.VTRK
000A    C.WRIT          0009    C.WRPR          F31E    CCA
F323    CCA1            F324    CCA2            00AF    CCS
00B1    CCS1            F57A    CDD             F584    CDD0
F592    CDD1            F59B    CDD2            0108    CFINIT
F6A5    CFT             F6A9I   CFTA            FB0E    CHECK
FD00    CHK00           FD20    CHK01           FD40    CHK02
FD60    CHK03           0000    CHK04           0000    CHK05
0000    CHK06           0000    CHK07           03F5    CHRDEL
03EA    CHRIN1          03F0    CHRIN2          03DC    CHRINS
0034    CHROM1          0035    CHROM2          FFB4    CHRSAV
FC55    CLOC            0350    CLR1            FD21    CLRBRK
001A    CLRCHR          0344    CLREOL          0361    CLREOS
0341    CLRLIN          001A    CLRS            0365    CLRS1
037A    CLRS2           0357    CLRSCN          0036    CMDSIZ
FAD8    CMDTAB          F767    CNFDPB          F000    COLD
F775    COMINP          F770    COMINS          F788    COMOTS
F77F    COMOUT          0518    COMRES          0518C   COMROM
F091    CONFG           F08B    CONFIG          F009    CONIN
F790    CONIOB          F00C    CONOUT          F006    CONST
```

FFBF	CONTBL	023D	CONTRL	F723	CPB
F731	CPB1	F736	CPB2	F759	CPB3
000D	CR	FC36	CRLF	0030	CRTBAS
0182	CRTD1	0196	CRTD2	0199	CRTD3
01B2	CRTD4	0169	CRTDVR	F2A3	CRTLDIR
3C00	CRTMAX	3000	CRTMEM	F2E7	CRTMV
F2E9	CRTMV0	F296	CRTOF1	F293	CRTOFF
F29C	CRTON	F299	CRTON1	F2F1	CRTOUT
FFE0	CRTSTK	003C	CRTTOP	0000	CSPACE
FFAE	CSRCHR	0018	CTC	0018	CTC0
0019	CTC1	001A	CTC2	001B	CTC3
FF10	CTCVEC	0036	CTLSIZ	0254	CTLTAB
FFAC	CURSOR	F6D7	CWP	FF56	DAY
F039	DAYTI	F086	DAYTIM	FDEB	DBL
F6F5	DCTRL	0000	DEBUG	02C5	DEFCUR
F6EA	DEFLPY	F6EB	DEFLUN	0017	DELLIN
FC80	DIRBUF	FFAF	DIRCUR	01F3	DIS1
01F2	DISATR	FDC2	DLN	FDC7	DLN1
FDCE	DLN2	FDD2	DLN3	FDE1	DLN4
FC55	DLOC	FC23	DMPFMT	02E7	DNCSR
FD72	DOC	F490	DPB5D	F470	DPB5S
F450	DPB8D	F430	DPB8S	F390	DPBASE
000A	DPBOFS	F470	DPBRG4	F480	DPBRG5
F490	DPBRG6	F4A0	DPBRG7	F1F1	DPM
F380	DRVTAB	FC76	DSK0	FC7B	DSK1
FCA0	DSK3	1367	DSKCMD	F4B0	DSKDVR
FCAC	DSKERR	03EF	DSM4	01EF	DSM5
00EF	DSM6	00EF	DSM7	0000	DSPACE
F632	DSW	FDB1	DTL	F6D5	DTYPE
FB8F	DUMP	FB98	DUMP2	FBB4	DUMP3
FBD1	DUMP4	FA08	DVRLMT	FD05	EATKEY
FC27	ECHO	F69B	ECR	F0A2	EIRET
01EF	ENATR	0000!	ENTRY	0004	EOT
00C4	ERR	00BC	ERR1	00C1	ERR2
00CC	ERR3	00D4	ERRM1	00DD	ERRM2
0009	ERRML	001B	ESC	029B	ESCADR
02BD	ESCAPE	028A	ESCTAB	0011	ESCTBL
FF34	EXPVEC	0000	FALSE	0000	FALUN
F2FE	FASTCRT	0001	FBLUN	0000	FCLUN
0002	FDLUN	FF30	FIFCNT	FF31	FIFIN
FF20	FIFO	FF32	FIFOUT	1428	FILL
FC59	FILL1	F6F6	FIRST	F708	FIRST1
F721	FIRST2	F4BD	FLOP1	F4DE	FLOP2
F4F2	FLOP3	F4FC	FLOP4	F506	FLOP5
F6EF	FLPFRM	0005	FM.DD	0007	FM.DDDS
00A0	FM.DDSS	0004	FM.DS	0006	FM.FV
0080	FM.HARD	0001	FM.SDDS	0000	FM.SDSS
0002	FM.SZ	0007	FM.UN	0003	FM.WR
0001	FMDD	0006	FMDDSS	0000	FMDS
001F	FORCE	02D4	FORSPC	F61A	GCA
F626	GCA0	F634	GCA1	F63F	GCA2
F627	GCAA	FE50	GCP	FE61	GCP1
FF1C	GENVEC	FBDA	GETHEX	FB2C	GETHLP
FB37	GETLIN	F097	GETSEL	FB38	GLIN1
FB50	GLIN4	FBDF	GNUM1	FBE4	GNUM3
F319	GOLD	12DB	GOTO	0009	GPIOCA

Balcones Operating System for the XEROX 820-II MACRO-80 3.44 09-Dec-81
The*End

000B	GPIOCB	0008	GPIODA	000A	GPIODB
177B	HELP	001E	HELPKEY	FBF3	HEXBIN
FA3C	HOME	02C9	HOMEUP	001E	HOMSCR
FF59	HRS	F649	ICC	F643	ICCS
F64D	ICCS1	F066	IDLE	FC5A	IN1
FC61	IN2	FC77	IN3	FC79	IN4
13CA	INCMD	F137	INDEX	0016	INSLIN
00E6	INTAB	FF50	INTSTK	112B	IOBDVR
008D	IOBDVS	F770	IOBLOC	0003	IOBYTE
F7AF	IOCONI	F796	IOCONO	F7A3	IOCONS
F7BB	IOLIST	F7CC	IOLSTS	FEEA	IPOINT
F643	ISC	FAD6	JPIX	001F	KBDCTL
001E	KBDDAT	F0D8	KBDIN	F0D5	KBDIN1
F0CD	KBDST	F0E2	KBMASK	0082	KDNLIN
F162	KEY1	F167	KEY2	F17F	KEY3
F184	KEY4	F18F	KEY5	F140	KEYSRV
0081	KUPLIN	FFBD	LAST	F5F0	LASTFM
FE0B	LBL	FE2E	LBL1	FD05	LCP
FD1D	LCP1	FD30	LCP2	FD52	LCP3
FD49	LCPA	FD4B	LCPB	F2B3	LDIR1
F2B5	LDIR2	F2C4	LDIR3	0418	LDIRX
FFB2	LEADIN	000A	LF	02F7	LFEED
FF5C	LINBUF	0388	LIND1	039E	LIND2
039F	LIND3	037C	LINDEL	03B7	LINI1
03C0	LINI2	03CD	LINI3	03A4	LININS
008A	LOCALF	0036	LOWLITE	F333	LSTATT
FD89	LTL	F6F1	LUN	0800	LX1984
021C	M3TST	022B	M4TST	0232	M5TST
F0E3	MASK	FB22	MDATA	FC5B	MDMP1
FC5E	MDMP2	FC63	MDMP3	FC69	MDMP3A
FC71	MDMP3B	12F2	MEMDMP	F236	MILO
F246	MILO1	F250	MILO2	F258	MILL1
F265	MILL2	F278	MILL3	F27C	MILL4
F27E	MILL5	F281	MILL6	F1FD	MILLI
FF50	MILSEC	FF5A	MINS	F5AF	MLU
01DF	MODE	01E1	MODE1	F000	MONITR
FF57	MONTH	00E5	MOVLN	F5C2	MPA
F5D7	MPA1	F5DA	MPA2	F5E6	MPA21
F5E9	MPA22	F606	MPA3	F610	MPA4
F613	MPA5	F559	MTRADR	01B4	MULTI
F6F4	NBLK	0066	NMI	02C0	NONO
0040	NT4	0020	NT5	0010	NT6
0010	NT7	0028	NTRK5	004D	NTRK8
F13F	NULINT	0006	NUMCON	F502	NUMUNT
4000	O.AUTO	0400	O.BAUD	0800	O.DDVR
0020	O.DISK	0010	O.ESCT	0004	O.FILL
2000	O.HELP	0200	O.INPC	0002	O.MOVE
0100	O.OUTC	1000	O.PROT	0040	O.RAMT
8000	O.RESV	0001	O.TERM	0008	O.TYPE
0080	O.VERF	F6F0	OPCODE	FEEC	OPOINT
BFFF	OPTIONS	13F1	OUTCMD	FE9B	OUTCRT
F288	OUTCUR	FC20	OUTPUT	0007	P.ACKN
0000	P.AUTO	0006	P.ONLN	0005	P.RDVI
0004	P.RDYO	0002	P.STRB	F59F	P2L
FB5D	PARA0	FB6C	PARA1	FB71	PARA2
FB76	PARA4	FFB5	PARAM1	FFB7	PARAM2

FFB9	PARAM3	FFBB	PARAM4	FB5F	PARAMS
000F	PASS8	FA0E	PHYCMD	FA15	PHYDMA
FA10	PHYDRV	FA13	PHYSEC	FA11	PHYTRK
FA0F	PHYUNT	F7F0	PIO1	0010	PIOAD
0011	PIOAS	0012	PIOBD	0013	PIOBS
F7DC	PIOOUT	F7F4	PIOSTO	FCB1	PKI
FCC9	PKI1	FCDF	PKI2	FCF3	PKI3
FCF6	PKI4	FCFE	PKI5	FD06	PKI6
FD0E	PKI7	FD14	PKI8	FD1B	PKI9
FC3D	PNEXT	FE78	PORTS	FD40	PRC
FAAC	PRMT1	FA62	PROMPT	FC68	PROT1
FC6F	PROT2	1459	PROTO	0000	PRS
0002	PRS1	003B	PRS2	0061	PRS3
0070	PRS4	00AC	PRS5	F339	PRVATT
FC1B	PUT2HS	FC03	PUT2HX	FB29	PUT2J
FC16	PUT4HS	FC0C	PUTNIB	FF00	RAM
041B	RBASE	FE6E	RCP	FE73	RCPA
F647	RDC	F605	RDID	F61B	RDID1
F6DC	RDONLY	F4E7	RDOP	FA4A	RDWR
F4FE	RDWRA	F4F6	RDWRS	FA48	READ
F641	RECAL	F6E9	RECLUN	F130	REMOVE
F6AD	RESET	0C55	RESLEN	F07C	RESTART
FC55	RESTOP	F1EF	RETINS	02F2	RETURN
F0A1	RETV1	F09A	RETVAL	F5F6	RETZR
0191	REV	F1E9	RFI	EE00	RGDBUF
0003	RGLUN	F6E8	RGRECAL	F708	RIGDPB
00B2	RMTALF	00B1	RMTTOG	0000	ROM
1800	ROMSIZ	17E1	ROMTOP	FA08	RQTOP
F5F8	RSE	F603	RSE1	F332	RSTATT
FFE2	RSTHL	FFE4	RSTPC	FFE0	RSTSP
0001	RTK4	0041	RTK5	0061	RTK6
0071	RTK7	1800	RX1984	0005	S.AUTOL
0004	S.AUTOR	0007	S.LECHO	0006	S.RECHO
F4B0	SA1403	F4C0	SAS0	F4B2	SAS0A
F4F3	SAS1	F4F6	SAS2	0012	SASIC
0010	SASID	0300	SASIDL	0012	SASIS
F470	SASSTR	F1EC	SAVSTK	0014	SCROLL
009E	SCRPRT	01C3	SEARCH	F4E6	SECLEN
FF5B	SECS	FA3E	SEEK	F5DE	SEEK0
F5EB	SEEK1	F5ED	SEEK2	F5EF	SEEK3
F5A3	SEEKX	0518!	SEGA	F5B9	SEK0
F5C4	SEK1	FA39	SEL1	F578	SEL1W
F585	SEL2	F591	SEL3	F598	SELDEN
F596	SELDNS	F510	SELEC	FA17	SELECT
F50B	SELER1	F42A	SELERR	F360	SELTAB
FA5A	SELTBL	F544	SELUNT	01D4	SETBLI
FD2C	SETBRK	021F	SETCOL	F0A4	SETCON
F284	SETCUR	01DD	SETGRA	01D9	SETINV
01CF	SETLOW	01F7	SETMSK	F337	SETPRV
0210	SETROW	0205	SETXY	0209	SETXY1
FC73	SIGN1	FC81	SIGN2	FC88	SIGN3
FC9E	SIGN3A	FCA6	SIGN4	FC55	SIGNON
F6CE	SIM	EF00	SIOBUF	0006	SIOCPA
0007	SIOCPB	0004	SIODPA	0005	SIODPB
FEE1	SIOI1	F0F0	SIOIN	F0ED	SIOIN1
FED6	SIOINC	FE7F	SIOINP	FE77	SIOINS

```
Balcones Operating System for the XEROX 820-II  MACRO-80 3.44   09-Dec-81
The*End

FEC8    SIOIST          F10C    SIOMSK          FE90    SIOOT
FE91    SIOOT1          F0F8    SIOOUT          FEA7    SIOPL
FEBC    SIOPL1          FEC5    SIOPL2          FEC6    SIOPL3
F113    SIORD1          F129    SIORD2          F12C    SIORD3
FE8A    SIORDT          F105    SIORDY          F0E5    SIOST
F10E    SIOVAL          FF00    SIOVEC          F0F9    SIOX1
0031    SLDDEN          FA08    SLERR           0030    SLSDEN
F65A    SMF             F66D    SMF0            F675    SMF0A
F693    SMF1            F6A3    SMF1A           F6BC    SMF1B
F6BE    SMF2            F6D4    SMF4            F6D1    SMFA
F50E    SMFS            F530    SMFS1           F548    SMFS1A
F550    SMFS2           F559    SMFS3           F573    SMFS4
F53D    SMFSA           03D1    SMP             FD4C    SNDLOC
FD5F    SNDRMT          F0D2    SOFT            F069    SOFTV
F6D2    SOM             F6D4    SOM1            FC1E    SPACE
F20E    SPACT           F22D    SPADDR          FFCD    SPARE
FFCB    SPARE1          FFCC    SPARE2          FFD0    SPARND
F224    SPCNT           F500    SSELEC          F0BF    SSP
0000    STACK           1070    START           FEE5    STATUS
F61D    STC             F644    STEPR           FF54    STEPRT
17E1    STOP            F66E    STPADR          02C1    STUFF
FCEE    SWAP            FCFB    SWAP1           0069    SYNC
001D    SYSCTL          001C    SYSPIO          FF18    SYSVEC
031F    TAB             F65F    TDI             F656    TDO
14E2    TERM            FC5E    TERM1           FC68    TERM2
FCA3    TERM3           13FB    TEST            FC5E    TEST1
FC61    TEST2           FC6D    TEST3           FF52    TIKCNT
F192    TIMER           F1B1    TIMER1          F1B9    TIMER2
F1E8    TIMER3          0006    TIMOU           FF55    TIMOUT
0771    TLOC            FEE8    TOPPTR          0299    TPAL
0299    TPAMAX          F6FF    TRKTBL          0100    TRMBUF
EF00    TRMSTK          F6ED    TRN5            F410    TRN6
FFFF    TRUE            F639    TTC             F63A    TTCA
FC5E    TYP0            FC90    TYP1            FCA4    TYP2
1477    TYPE            00FF    TYPTOG          02DC    UPCSR
F048    USRSEC          F31B    USRSTK          FF00    VECTAB
1443    VERCMD          FC5E    VERF1           FC66    VERF2
1315    VIEW            FC55    VIEW0           FC6F    VIEW1
FC89    VIEW2           FC8A    VIEW3           FC8E    VIEW4
FC90    VIEW5           F003    WARM            F35F    WASTED
F669    WCC             0010    WD1797          0010    WDCR
0031    WDDD            0013    WDDT            0030    WDSD
001C    WDSL            0012    WDSN            0010    WDSR
0011    WDTR            F687    WFR             F69A    WFR1
F696    WFRA            FAC9    WHAT            FE41    WLP
F64B    WOC             F64D    WOC1            F650    WOC2
FA44    WRITE           FE34    WUP             0000!   XCKS
0008!   XCKS1           0013    XOFF            F12D    XOFFLG
0011    XON             F115    XONENB          F344    XQDVR
FA51    XQPHYS          F508    XSELERR         FF58    YEAR
EE7D    Z.BAUA          EE7E    Z.BAUB          EE7F    Z.IOBT
EE62    Z.KEYM          EE5F    Z.SCRA          EE63    Z.SIOA
EE6D    Z.SIOB          EE77    Z.SIOM          EE79    Z.SIOV
EE60    Z.STPR          EE7B    Z.XONP


No Fatal error(s)
```

```
..        2433#  2450#  2450   2450   2450#  2450   2460#  2460   2460   2460#  2460   2470#
          2470   2470   2470#  2470   2480#  2480   2480   2480#  2480
.A        1756#
.B        1757#
.C        1758#
.D        1759#
.E        1761#
.F        1762#
.G        1763#
.H        1764#
.I        1766#
.J        1767#
.K        1768#
.L        1769#
.M        1771#
.N        1772#
.O        1773#
.P        1774#
ABORT       96#   815
ABOVE      177#   562    681   1701   1737   2415   3126   3142   3237   4872
ADDRH     2852   3035#
ADDRL     3036#
ALL00     1803#  1820
ALL01     1804#  1825
ALL02     1805#  1830
ALL03     1806#  1835
ALL04     1807#  1840
ALL05     1808#  1845
ALL06     1809#  1850
ALL07     1810#  1855
ASYNC       73#   526
ATTRIB    1179   1266   4952#
AUTOBT    3251#
AVAILB     446   4919#
AVAILT    4920#
B.BSY     2514#
B.CD      2516#  2943
B.IO      2518#
B.MSG     2515#  2922
B.PAR     2520#  2940
B.REQ     2517#  2938
B.RST     2521#  2978
B.SEL     2519#  2871
BAKSPC    1345   1445#
BASE      1132   1496   1508   1564   1571   1592   1623   4702   4950#
BAUD      3303   4047#
BAUD1     4053   4055#
BAUD2     4057   4059#
```

BAUDA	45#	523	3941	4056								
BAUDB	54#	507	3943	4058	4335							
BBASE	243	263#	266									
BBG	1503	1588	1616	1692#								
BELL	1344	1529#										
BELL1	1530#	1537										
BELLOF	66#	1533										
BELLON	67#	1530										
BELOW	171#	269	1162	4866								
BLOC	227#	271	563#	1164	1702#	3572	3663	3787	3958	3977	4003	4047
	4073	4135	4170	4190	4233	4254	4273	4299	4327	4387	4820	4861
	4868	4873#	4881									
BLOCAD	3554#	4259	4278									
BLOCK	3304	4254#										
BLOFC	102#	1534										
BLONC	101#	1531										
BLTIM	100#	1529										
BNDRY	1742#	1743	1744	1744								
BOOT	3302	3313	3787#									
BOOT1	3790#	3795										
BOOT2	3793	3801#										
BOOT3	3825	3834#										
BOOTBF	35#	132	3150	3834	3838	3902	3907	3910				
BOOTD	3803	3863#										
BOOTER	3808	3861#										
BOOTLD	34#	3837	3853									
BOOTS	3823	3855#	3899									
BOTPTR	4586	4595	4643	4652	4658	4662	4806#					
BRKFLG	4457#	4511	4513									
BRKKEY	4402#	4509										
BSEG	191#	239	270	1163	4867							
BSPACE	231#	270	270	270#	563	563	563#	682	682	1163	1163	1163#
	1702	1702	1702#	1738	1738	2416	2416	3127	3127	3143	3143	3238
	3238	3572	3572	3663	3663	3787	3787	3958	3958	3977	3977	4003
	4003	4047	4047	4073	4073	4135	4135	4170	4170	4190	4190	4233
	4233	4254	4254	4273	4273	4299	4299	4327	4327	4387	4387	4820
	4820	4861	4861	4867	4867	4867#	4873	4873	4873#	4878	4878	
BUFTOP	4410#	4679	4697									
C.BIN	2008#	2147	2187	2327								
C.FIVE	80#	3593	3596									
C.FLAW	2547#											
C.FLPY	2556#	3028										
C.FMAT	2544#	2887										
C.FTRK	2546#											
C.INIT	2552#											
C.KEYM	78#											
C.READ	2548#	2605	2747	3015								
C.RECAL	2541#	2862	3025									
C.RQSN	2543#											

C.RSYN	2542#											
C.SASI	79#	3601										
C.SEEK	2551#											
C.TRAM	2573#											
C.TRDY	2540#											
C.TWO	2009#											
C.VTRK	2545#											
C.WRIT	2550#	2603	2633									
C.WRPR	2549#											
CCA	957	1131#	1298	1615								
CCA1	1134#	1646										
CCA2	1135#	1136	1643									
CCS	328	359	370#									
CCS1	371#	378										
CDD	2691	2740#										
CDD0	2743	2745#										
CDD1	2750#	2753										
CDD2	2751	2754#										
CFINIT	437#	443										
CFT	2876	2935	2957#									
CFTA	2959#	2960	3054									
CHECK	3330#	4035	4214	4282								
CHK00	1794#	1820										
CHK01	1795#	1825										
CHK02	1796#	1830										
CHK03	1797#	1835										
CHK04	1798#	1840										
CHK05	1799#	1845										
CHK06	1800#	1850										
CHK07	1801#	1855										
CHRDEL	1398	1672#										
CHRIN1	1661#	1665										
CHRIN2	1660	1666#										
CHRINS	1400	1652#										
CHROM1	70#	1249										
CHROM2	71#	1242	1246									
CHRSAV	1168	1194	4953#									
CLOC	366	3268	3570#	3572	3572	3573	3663	3663	3663	3663	3664	3778
	3787	3787	3787	3788	3958	3958	3958	3958	3959	3977	3977	3977
	3978	4003	4003	4003	4004	4047	4047	4047	4048	4073	4073	4073
	4074	4135	4135	4135	4136	4170	4170	4170	4171	4190	4190	4190
	4191	4233	4233	4233	4234	4254	4254	4254	4255	4273	4273	4273
	4274	4299	4299	4299	4300	4327	4327	4327	4328	4387	4387	4387
	4388	4820	4820	4820	4820	4821	4861	4861	4861	4862	4867	5007
CLR1	1554#	1556										
CLRBRK	4459	4512#										
CLRCHR	4392#											
CLREOL	1361	1397	1546#	1569								
CLREOS	1354	1396	1569#									

CLRLIN	1504	1542#	1580	1607								
CLRS	90#	3618	4337	4427								
CLRS1	1571#	1581										
CLRS2	1578	1582#										
CLRSCN	1363	1410	1562#									
CMDSIZ	3328#											
CMDTAB	364	3260	3301#	3328								
CNFDPB	3057	3062	3075	3116#								
COLD	564#											
COMINP	595	3677#	3679	3725								
COMINS	594	3670#	3717									
COMOTS	597	3685	3693#	3744								
COMOUT	596	3685#	3686	3708	3734							
COMRES	234#	240	267#	270	563#	1163	1702#	4867	4873#			
COMROM	240#	270#	1163#	4867#								
CONFG	623#	3580										
CONFIG	584	620#										
CONIN	567#	3437	3440	3525	3656	4013	4147	4366	4460			
CONIOB	3701#	3707	3716	3724								
CONOUT	568#	3430	3433	3511	3528	3548	4014	4360	4363	4368	4373	
CONST	566#	3435	3654	4364	4448							
CONTBL	436	655	4965#									
CONTRL	1191	1325#										
CPB	3060	3074#										
CPB1	3077	3081#										
CPB2	3083#	3093										
CPB3	3102#	3113										
CR	86#	995	3244	3252	3353	3361	3386	3388	3438	3441	3537	3629
	3635	3643	3650	3792	3794	3796	4007	4026	4028	4150	4354	4361
	4369	4443	4538	4544	4550	4556	4827	4832	4836	4841	4847	
CRLF	3282	3341	3443	3536#	4090	4142						
CRTBAS	40#	1137	1465	1477	4708							
CRTD1	1179#	1315										
CRTD2	1173	1178	1191#									
CRTD3	1186	1190	1192#									
CRTD4	1203	1206#										
CRTDVR	1117	1167#										
CRTLDIR	581	1041#	3852	4089								
CRTMAX	39#	41	308									
CRTMEM	38#	39	40	305	307	308	385	426	428	1562		
CRTMV	1045	1077#										
CRTMVO	1071	1073	1078#									
CRTOF1	1020#											
CRTOFF	832	1018#	1078	1119	3278	3576						
CRTON	940	1010	1027#	1044	1115	3275						
CRTON1	1022#	1030										
CRTOUT	568	1091#										
CRTSTK	1042	1110	4975#									
CRTTOP	41#	1467	1475									

CSPACE	232#	270	563	563	682	682	1163	1163	1702	1702	1738	
	1738	2416	2416	3127	3127	3143	3143	3238	3238	3572	3572	3572#
	3663	3663	3663#	3663#	3787	3787	3787#	3787#	3958	3958	3958#	3958#
	3977	3977	3977#	3977#	4003	4003	4003#	4003#	4047	4047	4047#	4047#
	4073	4073	4073#	4073#	4135	4135	4135#	4135#	4170	4170	4170#	4170#
	4190	4190	4190#	4190#	4233	4233	4233#	4233#	4254	4254	4254#	4254#
	4273	4273	4273#	4273#	4299	4299	4299#	4299#	4327	4327	4327#	4327#
	4387	4387	4387#	4387#	4820	4820	4820#	4820#	4861	4861	4861#	4861#
	4867	4867	4867#	4873	4873	4878	4878					
CSRCHR	1195	1317	1320	4948#								
CTC	57#											
CTC0	58#	476										
CTC1	59#	480	675	991								
CTC2	60#	484										
CTC3	61#	488										
CTCVEC	409	478	4900#									
CTLSIZ	1370#											
CTLTAB	1330	1342#	1370									
CURSOR	425	1167	1192	4703	4947#							
CWP	2617	3004#										
DAY	615	4937#										
DAYTI	583#	2089	3052									
DAYTIM	583	615#										
DBL	4477	4640#										
DCTRL	2683	2735	3038#									
DEBUG	25#	29	30	123	128							
DEFCUR	1342	1434#										
DEFLPY	2837	3028#										
DEFLUN	2785	2828	3029#									
DELLIN	4391#											
DIRBUF	1793#	1819	1824	1829	1834	1839	1844	1849	1854			
DIRCUR	1005	1011	1014	4949#								
DIS1	1266#											
DISATR	1265#	1412										
DLN	4609#	4646										
DLN1	4613#	4615										
DLN2	4614	4616#										
DLN3	4620#	4635										
DLN4	4624	4630#										
DLOC	228#	564	682#	683	1163#	1703	1738#	1739	2416#	2417	3123#	3123
	3124	3127#	3128	3143#	3144	3238#	3239	3572#	4874	4878#	4881	
DMPFMT	3417	3419	3421	3515#								
DNCSR	1472#	1495	1579									
DOC	4533	4537	4543	4561#								
DPB5D	1943#											
DPB5S	1923#	3603										
DPB8D	1901#											
DPB8S	1882#	2388	2727									
DPBASE	1817#	2100	2677	3099								

DPBOFS	2011#	2109										
DPBRG4	1839	2450#	3095	3100								
DPBRG5	1844	2460#										
DPBRG6	1849	2470#										
DPBRG7	1854	2480#										
DPM	874	905#										
DRVTAB	1720	1782#										
DSK0	4080	4092#										
DSK1	4078	4095#										
DSK3	4113	4115#										
DSKCMD	3319	3324	4073#									
DSKDVR	1783	2018#	3048									
DSKERR	4109	4120#										
DSM4	2450#	2452	2453									
DSM5	2460#	2462	2463									
DSM6	2470#	2472	2473									
DSM7	2480#	2482	2483									
DSPACE	233#	270	563	563#	682	682#	682#	1163	1163#	1702	1702#	1738
	1738#	1738#	2416	2416#	2416#	3127	3127#	3127#	3143	3143#	3143#	3238
	3238#	3238#	3572	3572#	3663	3787	3958	3977	4003	4047	4073	4135
	4170	4190	4233	4254	4273	4299	4327	4387	4820	4861	4867	4873
	4873#	4878	4878#									
DSW	2169	2270#										
DTL	4489	4600#										
DTYPE	2324	2384#										
DUMP	3412#	3447	3996	4118								
DUMP2	3416#	3425										
DUMP3	3429#	3434										
DUMP4	3436	3443#										
DVRLMT	3129#											
EATKEY	3654#	3657										
ECHO	3357	3525#	4006	4025								
ECR	2915	2920	2923	2925	2949#							
EIRET	641#											
ENATR	1263#	1411										
ENTRY	241#	264	5025									
EOT	84#	3246	3296	3337	3376	3537	3546	3652	3867	3930	4123	4354
	4444	4472	4484	4641	4724	4849						
ERR	383	385#										
ERR1	318	320	382#									
ERR2	329	384#										
ERR3	388#	391										
ERRM1	382	394#	400									
ERRM2	384	397#										
ERRML	385	386	400#									
ESC	89#	3619	3926	3928	4339	4429	4472	4484	4641	4722		
ESCADR	1396#											
ESCAPE	1364	1422#										
ESCTAB	1222	1377#	1394									

ESCTBL	1223	1394#					
EXPVEC	4915#						
FALSE	23#	25					
FALUN	2525#	2762	2764				
FASTCRT	569	1096	1109#	3710	3738	4082	4755
FBLUN	2526#	2763					
FCLUN	2527#						
FDLUN	2528#	2765					
FIFCNT	417	689	769	814	4907#		
FIFIN	823	4908#					
FIFO	774	4906#					
FIFOUT	771	4909#					
FILL	3307	4233#					
FILL1	4238#	4244					
FIRST	2669	3045#	3050				
FIRST1	3057#	3066					
FIRST2	3061	3069#					
FLOP1	2025#	2080					
FLOP2	2042	2046	2048#				
FLOP3	2060	2063#					
FLOP4	2069#	2072					
FLOP5	2029	2035	2075#				
FLPFRM	2836	3031#					
FM.DD	2001#	2003	2370	2391			
FM.DDDS	2568#	2702	2765				
FM.DDSS	2003#	2325					
FM.DS	2000#	2379					
FM.FV	2002#	2329	2394				
FM.HARD	2569#	2766	2767	2768	2769	2805	3005
FM.SDDS	2566#	2763					
FM.SDSS	2565#	2762					
FM.SZ	2562#	2705					
FM.UN	1999#	2003	2366	2367	2371		
FM.WR	2563#						
FMDD	2561#	2703	2709				
FMDDSS	2567#	2764	2807				
FMDS	2560#	2686	2817				
FORCE	4394#	4626					
FORSPC	1349	1453#					
GCA	2860#	2889					
GCA0	2864	2866#					
GCA1	2873#	2877					
GCA2	2875	2878#					
GCAA	2866	2867#	3055				
GCP	4469	4481	4702#				
GCP1	4711#	4712					
GENVEC	4902#						
GETHEX	3395	3456#					
GETHLP	3351#	3359					

GETLIN	3249	3356#					
GETSEL	582	630#					
GLIN1	3357#	3369	3380				
GLIN4	3364	3373#					
GNUM1	3459#	3469					
GNUM3	3464#						
GOLD	1122#	1170	1276	1318	1498	1666	1682
GOTO	3308	3958#					
GPIOCA	51#	531					
GPIOCB	53#	536					
GPIODA	50#	3755					
GPIODB	52#	541	3756	3758	3760	3768	
HELP	3301	4820#					
HELPKEY	94#	3358	3526				
HEXBIN	3467	3476#	4016	4029			
HOME	574	3186#	3820				
HOMEUP	1367	1440#					
HOMSCR	4393#	4472	4484				
HRS	4940#						
ICC	2293	2298#					
ICCS	2622	2748	2838	2886#	2986		
ICCS1	2891#	2894					
IDLE	598#	702	722	732	2305	2957	
IN1	4142#	4159					
IN2	4145#	4149					
IN3	4151	4156#					
IN4	4153	4158#					
INCMD	3310	4135#					
INDEX	772#	824					
INSLIN	4390#						
INTAB	330	404#					
INTSTK	796	841	922	4926#			
IOBDVR	3663#	3848					
IOBDVS	447	3775#	3778	3850			
IOBLOC	447	2413#	3666	3775	3849		
IOBYTE	33#	3701	3732	3742	3898	3949	
IOCONI	590	3724#					
IOCONO	591	3707#					
IOCONS	589	3716#					
IOLIST	592	3732#					
IOLSTS	593	3742#					
IPOINT	4770	4778	4787	4809#			
ISC	2218	2291#	2344	2363			
JPIX	3289	3299#	3970				
KBDCTL	65#	469					
KBDDAT	64#	349	800				
KBDIN	567	703#	3727				
KBDIN1	702#	704					
KBDST	566	689#	703	3719			

KBMASK	708#	3681										
KDNLIN	4397#	4479										
KEY1	806	812#										
KEY2	803	814#										
KEY3	816	827#										
KEY4	829#	831										
KEY5	811	813	821	826	834#							
KEYSRV	406	794#										
KUPLIN	4396#	4467										
LAST	3983	3997	4961#									
LASTFM	2684	2831#	2832	2952								
LBL	4488	4657#										
LBL1	4667	4672#										
LCP	3847	3897#										
LCP1	3903	3908#										
LCP2	3915	3918#										
LCP3	3933#	3939										
LCPA	3922	3927#										
LCPB	3924	3929#										
LDIR1	1047#	1075										
LDIR2	1046	1048#										
LDIR3	1053	1055#										
LDIRX	1684	1695#										
LEADIN	1171	1325	4951#									
LF	85#	993	3244	3537	3629	3630	3635	3643	3650	4354	4362	4371
	4400	4443	4542	4554	4569	4575	4827	4832	4837	4841	4848	
LFEED	1189	1347	1489#									
LINBUF	1057	1065	1693	3247	3251	3283	3352	3354	3789	4076	4111	4580
	4672	4946#										
LIND1	1593#	1603	1605									
LIND2	1595	1606#										
LIND3	1607#	1638										
LINDEL	1399	1587#										
LINI1	1624#	1635										
LINI2	1627	1629#										
LINI3	1625	1636#										
LININS	1401	1613#										
LOCALF	4400#	4499										
LOWLITE	72#	1239										
LSTATT	1147#	1154										
LTL	4476	4574	4580#									
LUN	2784	3034#										
LX1984	32#	358	361									
M3TST	1288	1300#										
M4TST	1301	1312#										
M5TST	1313	1317#										
MASK	620	709#	1174	1274								
MDATA	3333	3341#	4005									
MDMP1	3983#											

MDMP2	3980	3984#										
MDMP3	3982	3987#										
MDMP3A	3991#	3993										
MDMP3B	3985	3996#										
MEMDMP	3305	3977#										
MESSAGE	4991#	5011	5012	5013	5014	5015	5016	5017	5018	5019	5020	5021
	5022											
MILO	944	953#										
MILO1	964#	969										
MILO2	968	970#										
MILL1	952	975#										
MILL2	937	982#										
MILL3	989	993#										
MILL4	984	995#										
MILL5	994	996#										
MILL6	932	935	981	998#								
MILLI	410	920#										
MILSEC	925	927	4930#									
MINS	4941#											
MLU	2609	2670	2778#									
MODE	1247	1250#										
MODE1	1244	1251#										
MONITR	37#	228	310	354	355	3572	3663	3787	3845	3958	3977	4003
	4047	4073	4135	4170	4190	4233	4254	4273	4299	4327	4387	4411
	4820	4861	4875	4881								
MONTH	4938#											
MOVLN	243	266#										
MPA	2620	2745	2801#									
MPA1	2808	2814#										
MPA2	2812	2816#										
MPA21	2821	2823#										
MPA22	2818	2826#										
MPA3	2835	2840#										
MPA4	2847#	2848										
MPA5	2806	2849#										
MTRADR	2091	2132	2133#	2152								
MULTI	1211#	1326										
NBLK	3037#											
NMI	1983#	2261	2264	2280								
NONO	1351	1352	1353	1355	1356	1357	1358	1359	1360	1362	1365	1366
	1424#											
NT4	2428#	2450	2450									
NT5	2429#	2460	2460									
NT6	2430#	2470	2470									
NT7	2431#	2480	2480									
NTRK5	2006#	2190										
NTRK8	2005#	2188										
NULINT	588	598	778#									
NUMCON	435	443#	648	4965								

NUMUNT	2653#											
O.AUTO	107#	128										
O.BAUD	111#	4041										
O.DDVR	110#	123	3783									
O.DISK	116#	124	4066									
O.ESCT	117#	123	128	1218	1341	4338	4428	4471	4483			
O.FILL	119#	125	4229									
O.HELP	108#	4815										
O.INPC	112#	4129										
O.MOVE	120#	3551	4250									
O.OUTC	113#	4164										
O.PROT	109#	4295										
O.RAMT	115#	126	4184									
O.RESV	106#	124										
O.TERM	121#	3637	4379									
O.TYPE	118#	3645	4323									
O.VERF	114#	125	3551	4269								
OPCODE	2606	2621	2632	2746	3014	3033#						
OPOINT	4775	4786	4801	4810#								
OPTIONS	123#	124#	124	125#	125	126#	126	128#	1218	1341	3551	3637
	3645	3783	4041	4066	4129	4164	4184	4229	4250	4269	4295	4323
	4338	4379	4428	4471	4483	4815						
OUTCMD	3316	4170#										
OUTCRT	4565	4627	4632	4754#								
OUTCUR	580	1010#										
OUTPUT	3500	3511#	4222									
OVERLAY	183#	3571	3662	3786	3957	3976	4002	4046	4072	4134	4169	4189
	4232	4253	4272	4298	4326	4386	4819	4860				
P.ACKN	146#											
P.AUTO	151#	542										
P.ONLN	147#											
P.RDYI	148#											
P.RDYO	149#	3770										
P.STRB	150#	542	3757	3759								
P2L	2762#	2778										
PARA0	3383#	3389										
PARA1	3390#	3403	3405									
PARA2	3394#											
PARA4	3397#											
PARAM1	3286	3397	4099	4957#								
PARAM2	3287	4958#										
PARAM3	3288	4959#										
PARAM4	3969	4116	4960#									
PARAMS	3284	3384#										
PASS8	4389#											
PHEX	2583#	3046										
PHYCMD	3145#	3223	3856									
PHYDMA	3150#	3215										
PHYDRV	3147#	3161	4100									

PHYSEC	3149#	3214										
PHYTRK	3148#	3193	3832									
PHYUNT	3146#											
PIO1	3762#	3763										
PIOAD	2502#	2506										
PIOAS	2501#	2502	2503	2504	2999							
PIOBD	2504#	2507	2508									
PIOBS	2503#	2975	2977									
PIOOUT	3735	3752#	3753									
PIOSTO	3745	3752	3768#									
PKI	4449	4456#										
PKI1	4462	4467#										
PKI2	4468	4479#										
PKI3	4478	4490#										
PKI4	4480	4491#										
PKI5	4492	4495#										
PKI6	4496	4499#										
PKI7	4500	4503#										
PKI8	4494	4498	4502	4506#								
PKI9	4504	4509#										
PMSG	2577#	3048										
PNEXT	3243	3292	3334	3374	3536	3542#	3549	3617	3861	3925	4120	4336
	4426	4470	4482	4640	4721	4822						
PORTS	4424	4517	4729	4730#								
PRC	4451	4530#										
PRINTX	4999#											
PRMT1	3267	3281#										
PROMPT	565	3241#	3253	3290	3297							
PROT1	4302	4312#										
PROT2	4315	4317#										
PROTO	3317	4299#										
PRS	276#	326	392	611								
PRS1	278#	279										
PRS2	311#	324										
PRS3	331#	340										
PRS4	342#	348										
PRS5	360	368#										
PRVATT	1153#	1155										
PUT2HS	3344	3424	3505#	3971	4144	4146						
PUT2HX	3488#	3503	3505									
PUT2J	3339	3344#										
PUT4HS	3342	3413	3502#	3972								
PUTNIB	3493	3495#										
RAM	36#	355	448	4894	4970	5007						
RBASE	353	4868#										
RCP	4490	4721#										
RCPA	4716	4723#										
RDC	2018	2149	2225	2294#	2354							
RDID	2234	2243#	2375									

RDID1	2251	2255#						
RDONLY	3008#	3068						
RDOP	2021	2056#						
RDWR	3200	3213#						
RDWRA	2063	2071#						
RDWRS	2065#	2199						
READ	576	3204#	3835	3908				
RECAL	2236	2286#	2334					
RECLUN	3026#							
REMOVE	706	769#						
RESET	2860	2950	2972#	2983				
RESLEN	4875#							
RESTART	564	607#						
RESTOP	4874#							
RETINS	827	901#						
RETURN	1350	1482#	1542	1692				
RETV1	636	640#						
RETVAL	616	626	634#					
RETZR	2228#	2323	2337					
REV	16#	625	3623	3623	3623			
RFI	835	896#	998					
RGDBUF	2497#	3074	3119					
RGLUN	2529#	2766	2767	2768	2769			
RGRECAL	2985	3025#						
RIGDPB	2412#	3602						
RMTALF	4399#	4503						
RMTTOG	4398#	4495						
ROM	29#	227	563	1702	4873	4887	4888	5003
ROMSIZ	30#	244	327	4887	4888	5003		
ROMTOP	4881#	4887	4888	5003				
RQTOP	3131#							
RSE	2213	2233#						
RSE1	2235	2238#						
RSTATT	1146#	1343						
RSTHL	281	4978#						
RSTPC	283	4979#						
RSTSP	280	4977#						
RTK4	2450#	2458						
RTK5	2460#	2468						
RTK6	2470#	2478						
RTK7	2480#	2488						
RX1984	31#	357	361	367				
S.AUTOL	4406#	4501	4540					
S.AUTOR	4407#	4505	4552					
S.LECHO	4404#	4464	4493					
S.RECHO	4405#	4497	4531					
SA1403	2595#	3058						
SASO	2604	2606#						
SASOA	2596#	2643						

SAS1	2634	2637#										
SAS2	2618	2636	2638#									
SASIC	2507#	2872	2879	2979	2981							
SASID	2506#	2870	2890									
SASIDL	3121#	3123	3604									
SASIS	2508#	2873	2921	2937								
SASSTR	2420#	3121										
SAVSTK	795	840	899#	921								
SCROLL	56#	304	1509	1565								
SCRPRT	95#	802										
SEARCH	1224#											
SECLEN	2631#	2801										
SECS	857	4942#										
SEEK	575	3192#										
SEEK0	2215#	2227										
SEEK1	2214	2222#										
SEEK2	2221	2223#										
SEEK3	2211	2225#										
SEEKX	2033	2178#										
SEGA	241#	271#	1164#	4868#								
SEGMENT	199#	270	563	682	1163	1702	1738	2416	3127	3143	3238	3572
	3663	3787	3958	3977	4003	4047	4073	4135	4170	4190	4233	4254
	4273	4299	4327	4387	4820	4861	4867	4873				
SEK0	2189	2191#										
SEK1	2185	2194	2199#									
SEL1	3160	3177	3181#									
SEL1W	2151#	2159	2163									
SEL2	2159#	2160	2162									
SEL3	2144	2148	2164#									
SELDEN	2130	2169#										
SELDNS	2168#	2180										
SELEC	2023	2085#										
SELECT	573	3158#	3811									
SELER1	2661#											
SELERR	1782	1873#	2087	2105								
SELTAB	365	630	1713	1754#	3608	3878						
SELTBL	3162	3231#										
SELUNT	2028	2120#	2322									
SERVICE	785#	794	839	920								
SETBLI	1242#	1405	1407									
SETBRK	4515	4517#										
SETCOL	1302#											
SETCON	586	646#										
SETCUR	579	1005#										
SETGRA	1249#	1406										
SETINV	1246#	1404										
SETLOW	1152	1239#	1403									
SETMSK	1271#	1408	1409									
SETPRV	1151#	1240	1251									

SETROW	1292#	1441				
SETXY	1283#	1402				
SETXY1	1215	1287#				
SIGN1	3591#	3591				
SIGN2	3595	3598#				
SIGN3	3582	3601#				
SIGN3A	3610#	3615				
SIGN4	3578	3600	3607	3617#		
SIGNON	368	3575#				
SIM	2749	2906	2913	2993#		
SIOBUF	4411#	4412	4774	4800	4809	4810
SIOCPA	48#	513	3670	3677	3693	4420
SIOCPB	49#	495	714	741	742	4423
SIODPA	46#	3680	3688	4420		
SIODPB	47#	725	735	949	996	4423
SIOI1	4799	4801#				
SIOIN	571	723#	754	3728	4358	
SIOIN1	722#	724				
SIOINC	4530	4795#	4796			
SIOINP	4735#	4736				
SIOINS	4729#	4735	4742	4764		
SIOIST	4450	4785#	4795			
SIOMSK	437	744#	3945	4308		
SIOOT	4526	4549	4555	4746#		
SIOOT1	4747#	4748				
SIOOUT	572	730#	3712	3737	4367	4372
SIOPL	4754	4757	4763#	4785		
SIOPL1	4773	4775#				
SIOPL2	4777	4779#				
SIOPL3	4765	4780#				
SIORD1	747	750#				
SIORD2	757	761#				
SIORD3	753	759	762#			
SIORDT	4742#	4747				
SIORDY	585	731	740#	934	3746	
SIOST	570	714#	723	752	3720	4356
SIOVAL	438	746#	3947	4311		
SIOVEC	499	4899#				
SIOX1	731#	733				
SLDDEN	69#					
SLERR	3133#					
SLSDEN	68#					
SMF	2103	2322#				
SMF0	2328	2330#				
SMF0A	2334#	2368				
SMF1	2348#	2353				
SMF1A	2351	2356#				
SMF1B	2365	2370#				
SMF2	2360	2371#				

SMF4	2376	2378	2383#									
SMFA	2126	2380	2381#									
SMFS	2655	2669#										
SMFS1	2689#	2704	2706									
SMFS1A	2701	2703#										
SMFS2	2695	2707#										
SMFS3	2710	2712#										
SMFS4	2699	2733#										
SMFSA	2689	2697#										
SMP	1593	1629	1642#									
SNDLOC	4465	4537#										
SNDRMT	4463	4532	4549#									
SOFT	599	695#										
SOFTV	599#	2079	2642									
SOM	2868	2998#										
SOM1	2994	2999#										
SPACE	3414	3426	3509#	3517								
SPACT	671	804	808	929#	976	983						
SPADDR	947#	975										
SPARE	4970#											
SPARE1	441	4967#										
SPARE2	442	4968#										
SPARND	4972#											
SPCNT	673	810	942#	945	971							
SSELEC	2599	2651#										
SSP	587	670#	812									
STACK	325	3242	4982#									
START	3572#											
STATUS	4446	4804#										
STC	2068	2244	2260#									
STEPR	2292#	2333										
STEPRT	440	3599	3920	4932#								
STOP	4861#											
STPADR	2093	2330	2331#									
STUFF	1368	1428#										
SWAP	3807	3876#										
SWAP1	3885#	3892										
SYNC	74#											
SYSCTL	63#	458	464	3585	3587							
SYSPIO	62#	461	608	610	851	853	938	980	1019	1022	1028	1048
	1061	1063	1074	1112	1254	1258	2510#	3272	3588	3592		
SYSVEC	405	471	4901#									
TAB	1346	1515#										
TDI	2635	2906#										
TDO	2637	2899#										
TERM	3309	4387#										
TERM1	4418	4420#										
TERM2	4422	4424#										
TERM3	4448#	4452										

TEST	3325	4190#										
TEST1	4199#	4223										
TEST2	4201#	4208										
TEST3	4211#	4219										
TIKCNT	844	846	4931#									
TIMER	412	839#										
TIMER1	850	854#										
TIMER2	858#	863										
TIMER3	860	867	882	888	892#							
TIMOU	2010#	2131	2155									
TIMOUT	848	4933#										
TLOC	229#	3572#	3572	3572	3663#	3663	3663	3787#	3787	3787	3958#	3958
	3958	3977#	3977	3977	4003#	4003	4003	4047#	4047	4047	4073#	4073
	4073	4135#	4135	4135	4170#	4170	4170	4190#	4190	4190	4233#	4233
	4233	4254#	4254	4254	4273#	4273	4273	4299#	4299	4299	4327#	4327
	4327	4387#	4387	4387	4820#	4820	4820	4861#	4861	4861	4881	
TOPPTR	4581	4585	4600	4604	4664	4668	4671	4807#				
TPAL	230#	3663	3663#	3787	3958	3958#	3977	4003	4047	4073	4135	4170
	4190	4233	4254	4273	4299	4327	4387	4820	4820#	4861	4862	4867
	5007											
TPAMAX	3270	4862#										
TRKTBL	2122	2408#										
TRMBUF	4409#	4410	4684	4691	4806	4807						
TRMSTK	4412#	4425										
TRN5	2393	2402#										
TRN6	1865#	2396	2711									
TRUE	22#	23										
TTC	2073	2249	2278#									
TTCA	2262	2279#										
TYP0	4330	4336#										
TYP1	4356#	4365	4370	4374								
TYP2	4357	4364#										
TYPE	3321	4327#										
TYPTOG	4401#	4491										
UPCSR	1348	1462#										
UPDATE	208#	270	563	682	1163	1702	1738	2416	3127	3143	3238	3572
	3663	3787	3958	3977	4003	4047	4073	4135	4170	4190	4233	4254
	4273	4299	4327	4387	4820	4861	4867	4873	4877			
USRSEC	588#	847										
USRSTK	1041	1079	1109	1123	1124#	1501						
VECTAB	351	4898#										
VERCMD	3323	4273#										
VERF1	4281#	4289										
VERF2	4279	4287#										
VIEW	3306	3314	4003#									
VIEW0	4005#	4039										
VIEW1	4012	4016#										
VIEW2	4027	4033#										
VIEW3	4015	4034#										

VIEW4	4008	4036#									
VIEW5	4010	4038#									
WARM	565#	833	3655	3846							
WASTED	1741#										
WCC	2754	2839	2900	2902	2908	2913#	2987				
WD1797	55#	3590	3598								
WDCR	1988#	2267	2298	2346							
WDDD	1994#	2171	2338								
WDDT	1991#	2216	2266	2342							
WDSD	1993#	2173	2361								
WDSL	1992#	2136	2141	2186	2203	2207	2326	2372	2374		
WDSN	1990#	2048									
WDSR	1987#	2157	2164	2306	2349						
WDTR	1989#	2209	2253								
WFR	2750	2891	2899	2907	2914	2919	2935#	2939			
WFR1	2936	2941	2945#								
WFRA	2942#										
WHAT	3250	3256	3292#	3311	3312	3315	3318	3320	3322	3326	3327
WLP	4603	4661	4670	4689#							
WOC	2282	2302#									
WOC1	2303#	2304									
WOC2	2305#	2308									
WRITE	577	3199#									
WUP	4584	4594	4651	4678#							
XCKS	243#										
XCKS1	246#	253									
XOFF	88#	756	758								
XOFFLG	761	763#									
XON	87#	758									
XONENB	439	751#	3918	4317							
XQDVR	578	1709#	3225								
XQPHYS	3173	3223#	4106	4115							
XSELERR	2660#										
YEAR	4939#										
Z.BAUA	140#	141	3940								
Z.BAUB	141#	142	3942								
Z.IOBT	142#	3948									
Z.KEYM	134#	135	3923								
Z.SCRA	132#	133	3921								
Z.SIOA	135#	136	3931								
Z.SIOB	136#	137									
Z.SIOM	137#	138	3944								
Z.SIOV	138#	139	3946								
Z.STPR	133#	134	3919								
Z.XONP	139#	140	3913								

Notes

```
Quick, Fast Cold Start Loader   MACRO-80 3.44   09-Dec-81

                                   Title   Quick, Fast Cold Start Loader
                          ;;;      Quick, Fast Cold Start Loader.
                          ;
                          ;        Copyright (C) 1982, Balcones Computer Corporation.
                          ;
                                   .z80
        F02A              Xqdvr    equ     0f02ah          ;Physical Driver Executioner
        0004              cdisk    equ     00004h          ;current user/disk
        0000'                      cseg
        0000'             bios     equ     $               ;origin of bios above ccp & bdos
                                   .phase  80h
        0080              phycmd   equ     $
        0081              phyunt   equ     $+1
        0082              phydrv   equ     $+2
        0083              phytrk   equ     $+3
        0085              physec   equ     $+5
        0087              phydma   equ     $+7
                          ;;       quick, fast loader.
                          ;
                          ;        Entry:  A =  Sectors per Track
                          ;                DE = Address of Physical Command Block that loaded QFD
                          ;
        0080    EB        qfs:     ex      de,hl           ;transfer command block
        0081    01 0005            ld      bc,physec-phycmd
        0084    ED B0              ldir                    ;woe be unto he who changes qfs
        0086    21 00D5            ld      hl,ldrtbl-5     ;set loader control table address
        0089    0E 05              ld      c,5             ;set table entry size
        008B    09        qfsl:    add     hl,bc           ;advance table address
        008C    BE                 cp      (hl)            ;match with loader control table entry
        008D    38 FC              jr      c,qfsl          ;if match not found yet
        008F    C0                 ret     nz              ;if entry not in table
        0090    23                 inc     hl
        0091    4E                 ld      c,(hl)          ;set track offset+1
        0092    23                 inc     hl
        0093    ED A0              ldi                     ;move starting sector, adjust track offset
        0095    AF                 xor     a
        0096    12                 ld      (de),a          ;clear upper sector
        0097    7E                 ld      a,(hl)          ;set number of sectors
        0098    23                 inc     hl
        0099    6E                 ld      l,(hl)          ;set (sector size)/4-1
        009A    23                 inc     hl              ;sector size / 4
        009B    29                 add     hl,hl
        009C    29                 add     hl,hl
        009D    EB                 ex      de,hl           ;set sector size in DE
        009E    2A 0083            ld      hl,(phytrk)     ;add track offset
        00A1    09                 add     hl,bc
        00A2    22 0083            ld      (phytrk),hl
        00A5    47                 ld      b,a             ;set number of sectors
        00A6    7A                 ld      a,d             ;check sector size
        00A7    21 FF80'           ld      hl,bios-80h     ;set starting address
```

```
00AA    B7                      or      a
00AB    20 01                   jr      nz,qfs3         ;if big sectors
00AD    19              qfs2:   add     hl,de           ;advance transfer address
00AE    22 0087         qfs3:   ld      (phydma),hl
00B1    D9                      exx                     ;switch register sets
00B2    21 0080                 ld      hl,phycmd or qfs;set physical command address
00B5    CD F02A                 call    Xqdvr           ;execute driver read
00B8    21 0085                 ld      hl,physec       ;advance sector
00BB    34                      inc     (hl)
00BC    D9                      exx                     ;switch back
00BD    B7                      or      a
00BE    C0                      ret     nz              ;if boot error
00BF    10 EC                   djnz    qfs2            ;if cold start not complete
00C1    32 0004                 ld      (cdisk),a       ;start out in A: user 0
00C4    21 0003'                ld      hl,bios+3       ;warm start after signon
00C7    E5                      push    hl
00C8    21 00F6                 ld      hl,sernum
00CB    11 00DA                 ld      de,ldrtbl
00CE    0E 0A                   ld      c,sernml
00D0    ED B0                   ldir
00D2    12                      ld      (de),a
00D3    13                      inc     de
00D4    3E C9                   ld      a,0c9h
00D6    12                      ld      (de),a
00D7    CD 0000'                call    bios            ;execute cold start loader

                        ;;      Loader Control Table.
                        ;
                        ;       Entries Must be in sort order
                        ;
                        ;       db      sectors per track
                        ;               offset+1 from boot track
                        ;               starting bios sector
                        ;               number of bios sectors
                        ;               sector size/4-1
                        ;
00DA    34 02 16 04     ldrtbl: db      52,1+1,22,4,256/4-1     ;8" Double Density
00DE    3F
00DF    22 03 05 04             db      34,2+1,05,4,256/4-1     ;5" Double Density
00E3    3F
00E4    1A 02 14 07             db      26,1+1,20,7,128/4-1     ;8" Single Density
00E8    1F
00E9    12 03 0A 07             db      18,2+1,10,7,128/4-1     ;5" Single Density
00ED    1F
00EE    00 01 16 04             db      00,0+1,22,4,256/4-1     ;Any SASI Rigid Disk
00F2    3F
00F3    00                      db      00                      ;    End of table

                                if      $ gt 100h-10
                                .printx * Too Big *
                                else
                                if      $ eq 100h-10
                                .printx + Perfect Fit +
                                else
00F4                            ds      100h-10-$,-1
                                endif
```

```
Quick, Fast Cold Start Loader   MACRO-80 3.44   09-Dec-81

                                          endif

                                  ;;      Serialization.
                                  ;
      00F6    20 44 43 2A         sernum: db        DC*******'          ;Manufacturing serial number here
      00FA    2A 2A 2A 2A
      00FE    2A 2A
      000A                        sernml  equ     $-sernum

                                          if      $ ne 100h
                                          .printx * Serial Number Out of Place *
                                          endif
                                          .dephase

                                          end
```

```
Quick, Fast Cold Start Loader   MACRO-80 3.44   09-Dec-81

Macros:

Symbols:
0000'   BIOS            0004    CDISK           00DA    LDRTBL
0080    PHYCMD          0087    PHYDMA          0082    PHYDRV
0085    PHYSEC          0083    PHYTRK          0081    PHYUNT
0080    QFS             008B    QFS1            00AD    QFS2
00AE    QFS3            000A    SERNML          00F6    SERNUM
F02A    XQDVR


No Fatal error(s)
```

BIOS	13#	56	71	81
CDISK	10#	70		
LDRTBL	32	74	93#	
PHYCMD	17#	30	62	
PHYDMA	22#	60		
PHYDRV	19#			
PHYSEC	21#	30	64	
PHYTRK	20#	51	53	
PHYUNT	18#			
QFS	29#	62		
QFS1	34#	36		
QFS2	59#	69		
QFS3	58	60#		
SERNML	75	120#		
SERNUM	73	117#	120	
XQDVR	9#	63		

Notes

```
                                      subttl  Bios Jump Table
                                      title   XEROX 820-II BIOS

                              ;;      XEROX 820+ Rom Resident Bios Jump Table.
                              ;
                              ;       Copyright 1981, Balcones Computer Corporation.
                              ;
                                      .z80

        0000'   C3 00F1'      bios:   jp      cboot           ;cold start
        0003'   C3 0069'      bwboot: jp      wboot           ;warm start

        0006'   C3 F04B       bconst: jp      const           ;console status
        0009'   C3 F04E       bconin: jp      conin           ;console character in
        000C'   C3 F051       bconot: jp      conout          ;console character out
        000F'   C3 F054       bprint: jp      list            ;list character on printer
        0012'   C3 F060       bpunch: jp      punch           ;punch
        0015'   C3 F05D       breadr: jp      reader          ;reader

        0018'   C3 01B6'      bhome:  jp      home            ;move head to hcme position
        001B'   C3 0154'      bseld:  jp      seldsk          ;select disk
        001E'   C3 01B9'      bsett:  jp      settrk          ;set track number
        0021'   C3 01BE'      bsets:  jp      setsec          ;set sector number
        0024'   C3 01C3'      bsetd:  jp      setdma          ;set dma address
        0027'   C3 01EB'      bread:  jp      read            ;read a record
        002A'   C3 01F3'      bwrit:  jp      write           ;write a record

        002D'   C3 F057       bprnts: jp      listst          ;printer ready status
        0030'   C3 01C8'      bsctrn: jp      sectrn          ;sector translate

        0033'   81            initio: db      10000001b       ;Initial I/O Byte

                                      Subttl  Cold and Warm Start Module
                                      page
```

```


        0E00                    bdosln  equ     0e00h           ;Length of CP/M v 2.2 BDOS
        0800                    ccplen  equ     0800h           ;Length of CP/M v 2.2 CCP
        0004                    cdisk   equ     4               ;Current user/disk
        002C                    nsects  equ     (ccplen+bdosln)/128     ;number of sectors for ccp + bdos
        0062                    rev     equ     'b'

                                ;;      Wboot - Warm Start CP/M.
                                ;
        0034'   3E C3           wbt5:   ld      a,0c3h          ;plant jumps
        0036'   21 F206'                ld      hl,bios-bdosln+6
        0039'   BE                      cp      (hl)
        003A'   20 1C                   jr      nz,wbterr       ;if no jump to bdos
        003C'   32 0000                 ld      (0),a
        003F'   32 0005                 ld      (5),a
        0042'   22 0006                 ld      (6),hl          ;set address of jump to bdos
        0045'   21 0003'                ld      hl,bwboot       ;set warm boot address
        0048'   22 0001                 ld      (1),hl
        004B'   ED 4B 0004              ld      bc,(cdisk)      ;set current disk / user
        004F'   21 EA00'                ld      hl,bios-bdosln-ccplen ;Enter CCP
        0050'                   wbtcom  equ     $-2             ;patch to "03" to disable warm boot command
        0052'   3E 03                   ld      a,3
        0054'   32 0050'                ld      (wbtcom),a
        0057'   E9                      jp      (hl)

        0058'   CD 0115'        wbterr: call    pmsg            ;display error message
        005B'   0D 0A 42 6F             db      13,10,'Boot Err',0
        005F'   6F 74 20 45
        0063'   72 72 00
        0066'   CD 0009'                call    bconin  ;wait for key
        0069'   31 0100         wboot:  ld      sp,100h         ;use external stack
        006C'   CD 013F'                call    dboot           ;inform deblocker
        006F'   4F                      ld      c,a             ;(zero) select A:
        0070'   3E 2C                   ld      a,nsects        ;set number of sectors to read
        0072'   32 00D0'                ld      (seccnt),a      ;set sector counter
        0075'   21 E980'                ld      hl,bios-bdosln-ccplen-128
        0078'   22 013B'                ld      (dmabas),hl     ;set base track dma address
        007B'   CD 001B'                call    bseld           ;select boot drive (A:)
        007E'   7C                      ld      a,h
        007F'   B5                      or      l
        0080'   28 D6                   jr      z,wbterr
        0082'   23                      inc     hl              ;point to high translate address
        0083'   7E                      ld      a,(hl)
        0084'   32 0122'                ld      (xlate),a
        0087'   E5                      push    hl
        0088'   0E 00                   ld      c,0             ;translate sector zero
        008A'   CD 0121'                call    mls
        008D'   79                      ld      a,c
        008E'   32 0133'                ld      (transz),a      ;set sector zero translate value
        0091'   E1                      pop     hl
        0092'   11 0009                 ld      de,10-1         ;offset to dpb
        0095'   19                      add     hl,de
        0096'   4E                      ld      c,(hl)          ;get dpb address
        0097'   23                      inc     hl
```

```
0098'   46                      ld      b,(hl)
0099'   0A                      ld      a,(bc)          ;get low sectors per track
009A'   32 00DC'                ld      (spt),a
009D'   21 000D                 ld      hl,13
00A0'   09                      add     hl,bc
00A1'   4E                      ld      c,(hl)          ;get reserved tracks
00A2'   23                      inc     hl
00A3'   46                      ld      b,(hl)
00A4'   0B                      dec     bc
00A5'   1E 01                   ld      e,1             ;set sector 1
00A7'   B7                      or      a               ;test low sectors per track
00A8'   28 06                   jr      z,wbt1          ;if rigid disk
00AA'   4A                      ld      c,d             ;set track 0
00AB'   FE 1B                   cp      26+1
00AD'   38 01                   jr      c,wbt1          ;if single density 8" or 5"
00AF'   4B                      ld      c,e             ;double density starts on track 1, sector 1
00B0'   C5              wbt1:   push    bc              ;save track
00B1'   D5                      push    de              ;save starting sector
00B2'   CD 001E'                call    bsett           ;position disk
00B5'   C1                      pop     bc
00B6'   C5              wbt2:   push    bc              ;save sector
00B7'   CD 0121'                call    mls             ;map logical sector
00BA'   E5                      push    hl              ;save address
00BB'   CD 0021'                call    bsets           ;set sector
00BE'   C1                      pop     bc
00BF'   21 FFFE'                ld      hl,bios-2
00C2'   ED 42                   sbc     hl,bc
00C4'   38 12                   jr      c,wbt3          ;if within bios
00C6'   CD 0024'                call    bsetd           ;set dma address
00C9'   CD 01EB'                call    read            ;read next sector
00CC'   B7                      or      a
00CD'   20 89                   jr      nz,wbterr       ;if load error
00CF'   3E 00                   ld      a,0             ;update sectors read counter
00D0'                   seccnt  equ     $-1
00D1'   3D                      dec     a
00D2'   32 00D0'                ld      (seccnt),a
00D5'   CA 0034'                jp      z,wbt5          ;if end of load
00D8'   C1              wbt3:   pop     bc              ;advance sector
00D9'   0C                      inc     c
00DA'   79                      ld      a,c
00DB'   FE FF                   cp      -1
00DC'                   spt     equ     $-1
00DD'   20 D7                   jr      nz,wbt2         ;if not end of track
00DF'   41                      ld      b,c
00E0'   2A 013B'                ld      hl,(dmabas)     ;advance base dma address
00E3'   11 0080                 ld      de,128
00E6'   19              wbt4:   add     hl,de           ;by spt*128
00E7'   10 FD                   djnz    wbt4
00E9'   22 013B'                ld      (dmabas),hl
00EC'   C1                      pop     bc              ;advance track
00ED'   0C                      inc     c
00EE'   58                      ld      e,b             ;and restart on sector 0
00EF'   18 BF                   jr      wbt1

                        ;;      Cold Start CP/M.
                        ;
```

```
00F1'   CD 0115'            cboot:  call    pmsg            ;Announce CP/M size and version
00F4'   1A 58 65 72                 db      26,'Xerox 60k CP/M vers 2.2',rev
00F8'   6F 78 20 36
00FC'   30 6B 20 43
0100'   50 2F 4D 20
0104'   76 65 72 73
0108'   20 32 2E 32
010C'   62
010D'   20 23 32 2D                 db        #2-294'
0111'   32 39 34
0114'   00                          db      0

                            ;;      pmsg - print message at return address.
                            ;
0115'   E1                  pmsg:   pop     hl              ;print message after call
0116'   7E                          ld      a,(hl)
0117'   23                          inc     hl
0118'   B7                          or      a
0119'   E5                          push    hl
011A'   C8                          ret     z               ;if end of message
011B'   4F                          ld      c,a
011C'   CD 000C'                    call    bconot          ;display message at current console
011F'   18 F4                       jr      pmsg

                            ;;      mls - map logical sector.
                            ;
0121'   3E 00               mls:    ld      a,0             ;set translate address
0122'                       xlate   equ     $-1
0123'   B7                          or      a
0124'   79                          ld      a,c
0125'   28 0B                       jr      z,mls2          ;if not single density
0127'   87                          add     a,a             ;read by half tracks
0128'   2A 00DC'                    ld      hl,(spt)        ;get sectors per track
012B'   BD                          cp      l
012C'   38 02                       jr      c,mls1          ;if not past end of track
012E'   95                          sub     l               ;offset back to beginning of track
012F'   3C                          inc     a
0130'   3C                  mls1:   inc     a               ;map sector 0->1
0131'   4F                          ld      c,a
0132'   D6 00               mls2:   sub     0               ;offset by translate of sector zero
0133'                       transz  equ     $-1
0134'   1F                          rra
0135'   67                          ld      h,a
0136'   2E 00                       ld      l,0
0138'   CB 1D                       rr      l
013A'   11 0000                     ld      de,0            ;set base dma for this track
013B'                       dmabas  equ     $-2
013D'   19                          add     hl,de           ;compute address for this sector
013E'   C9                          ret

                                    Subttl  CHARIO - Character I/O Module
                                    page
```

```
        F000                    monitr  equ     0f000h          ;820+ Resident Monitor Address

        F04B                    const   equ     monitr+4bh
        F04E                    conin   equ     monitr+4eh
        F051                    conout  equ     monitr+51h
        F054                    list    equ     monitr+54h
        F057                    listst  equ     monitr+57h
        F05D                    reader  equ     monitr+5dh
        F060                    punch   equ     monitr+60h

                                        Subttl  CP/M Deblocking Driver
                                        page
```

```


                                ;;      CP/M Deblocking Driver.
                                ;
013F'                                   cseg

                                ;;      Ascii.
                                ;
000A                            lf      equ     10
000B                            up      equ     11
000D                            cr      equ     13
001B                            esc     equ     27

451B                            inslin  equ     ('E' shl 8) + esc
521B                            dellin  equ     ('R' shl 8) + esc

                                ;;      Absolute Machine Addresses.
                                ;
F02A                            xqdvr   equ     0f02ah          ;Resident Monitor Driver Executioner
0004                            cdisk   equ     4               ;CCP active user/disk

                                ;;      CP/M Write Types.
                                ;
0000                            wrall   equ     0               ;normal write to allocated sector
0001                            wrdir   equ     1               ;write to directory sector
0002                            wrual   equ     2               ;first write to unallocated block

                                ;;      skip - skip next instruction.
                                ;
                                ;       Uses HL to perform very short jumps
                                ;
                                skip    macro   n
                                        if      ((n)-$) eq 2
                                        db      26h             ;;;set PC = $+2  (ld h,...)
                                        endif
                                        if      ((n)-$) eq 3
                                        db      21h             ;;;set PC = $+3  (ld hl,...)
                                        endif
                                        endm

                                ;;      Dboot - Deblocking Bootstrap.
                                ;
                                ;       Entry:  Called prior to Warm Start reload.
                                ;
013F'   21 0000"                dboot:  ld      hl,hstbuf       ;initialize host buffer address
0142'   22 021A"                        ld      (hstdma),hl
0145'   21 021C"                        ld      hl,dphtab       ;clear internal DPH table of addresses
0148'   01 2000                         ld      bc,16*2*256     ;set table length, zero
014B'   71                      dbt2:   ld      (hl),c          ;clear next byte
014C'   23                              inc     hl
014D'   10 FC                           djnz    dbt2            ;if table not clear

                                ;;      clract - Clear host buffer active.
                                ;
014F'   AF                      clract: xor     a
```

```
0150'   32 026E'                 ld      (hstact),a       ;clear host buffer active
0153'   C9                       ret

                        ;;      select - select CP/M disk.
                        ;
                        ;       Entry:  C = CP/M Logical Drive, 0-15.
                        ;               E = 2*n+0 if media identification required
                        ;               E = 2*n+1 if media previously identified
                        ;
0154'   79              seldsk:  ld      a,c              ;remember disk to seek
0155'   32 0205"                 ld      (sekdsk),a
0158'   06 00                    ld      b,0
015A'   21 021C"        sel1:    ld      hl,dphtab        ;set table of remembered dph's
015D'   09                       add     hl,bc            ;index by words
015E'   09                       add     hl,bc
015F'   CB 43                    bit     0,e
0161'   28 0A                    jr      z,sel2           ;if drive not previously selected
0163'   7E                       ld      a,(hl)           ;set disk parameter header address in hl
0164'   23                       inc     hl
0165'   66                       ld      h,(hl)
0166'   6F                       ld      l,a
0167'   B4                       or      h
0168'   20 19                    jr      nz,sel3          ;if previous select succesful
016A'   5F                       ld      e,a              ;force media identification
016B'   18 ED                    jr      sel1
016D'   E5              sel2:    push    hl               ;save dph table address
016E'   21 0202"                 ld      hl,selcmd+2
0171'   71                       ld      (hl),c           ;set CP/M Logical drive
0172'   2B                       dec     hl
0173'   2B                       dec     hl               ;point to select command
0174'   36 FF                    ld      (hl),-1          ;set driver select operation
0176'   CD 02DF'                 call    xdr              ;execute driver request
0179'   EB                       ex      de,hl
017A'   E1                       pop     hl
017B'   73                       ld      (hl),e           ;remember disk parameter header address
017C'   23                       inc     hl
017D'   72                       ld      (hl),d
017E'   EB                       ex      de,hl
017F'   7D                       ld      a,l
0180'   B4                       or      h
0181'   28 22                    jr      z,sel4           ;if drive not succesfully selected
0183'   E5              sel3:    push    hl               ;save dph address
0184'   01 000A                  ld      bc,10            ;set dpb offset in dph
0187'   09                       add     hl,bc
0188'   5E                       ld      e,(hl)           ;set disk parameter block address
0189'   23                       inc     hl
018A'   56                       ld      d,(hl)
018B'   EB                       ex      de,hl
018C'   22 0241'                 ld      (dpbadr),hl
018F'   0E 03                    ld      c,3
0191'   09                       add     hl,bc
0192'   7E                       ld      a,(hl)           ;set block shift factor
0193'   3C                       inc     a                ;form 128 byte records per block
0194'   32 021B'                 ld      (rpb),a
0197'   0E 0C                    ld      c,15-3           ;point to end of dpb
0199'   09                       add     hl,bc
```

```
019A'   7E                      ld      a,(hl)          ;set internal parameter
019B'   32 0208'                ld      (trkzfl),a      ;set track zero single density flag
019E'   E6 07                   and     7
01A0'   32 029B'                ld      (secmsk),a      ;set sector mask
01A3'   E1                      pop     hl
01A4'   C9                      ret
01A5'                   sel4:                           ;fall into clear active disk

                        ;;      cad - Clear Active Disk.
                        ;
01A5'   21 0004         cad:    ld      hl,cdisk        ;get disk that CCP will log in
01A8'   3A 0205"                ld      a,(sekdsk)      ;get disk that failed
01AB'   AE                      xor     (hl)
01AC'   E6 0F                   and     not 11110000b   ;clear active user
01AE'   20 04                   jr      nz,cad1         ;if selected disk is not default disk
01B0'   7E                      ld      a,(hl)          ;cause CCP to log in A:
01B1'   E6 F0                   and     not 1111b       ;retain active user area
01B3'   77                      ld      (hl),a
01B4'   6C              cad1:   ld      l,h             ;indicate select failure
01B5'   C9                      ret

                        ;;      Home - Set Track Zero.
                        ;
01B6'   01 0000         home:   ld      bc,0            ;seek track zero

                        ;;      Settrk - Set Track.
                        ;
                        ;       Entry:  BC = Track number
                        ;
01B9'   ED 43 0206"     settrk: ld      (sektrk),bc     ;set track to seek
01BD'   C9                      ret

                        ;;      Setsec - Set Sector.
                        ;
                        ;       Entry:  BC = Sector number
                        ;
01BE'   ED 43 020C"     setsec: ld      (seksec),bc     ;set sector to seek
01C2'   C9                      ret

                        ;;      Setdma - Set Direct Memory Address.
                        ;
                        ;       Entry:  BC = DMA address
                        ;
01C3'   ED 43 020A"     setdma: ld      (sekdma),bc
01C7'   C9                      ret

                        ;;      Sectran - Sector Translate.
                        ;
                        ;       Entry:  BC = Sector number, 0 <= BC <  Sectors per Track
                        ;               DE = Single byte skew table address
                        ;
                        ;       Exit:   HL = BC         if DE =  0
                        ;                L = (DE+BC)    if DE <> 0
                        ;                H = B          which better be zero
                        ;
01C8'   69              sectrn: ld      l,c             ;set untranslated sector
```

```
01C9'   60                      ld      h,b
01CA'   7A                      ld      a,d
01CB'   B3                      or      e
01CC'   C8                      ret     z               ;if no translate table
01CD'   EB                      ex      de,hl
01CE'   09                      add     hl,bc
01CF'   6E                      ld      l,(hl)          ;single byte translate
01D0'   60                      ld      h,b
01D1'   C9                      ret

                        ;;      Rdwrs - Read or Write Single Density.
                        ;
01D2'   3A 02AE'        rdwrs:  ld      a,(readop)      ;set read/write operation
01D5'   21 0203"                ld      hl,sekcmd       ;set seek request
01D8'   18 07                   jr      rdwrhs          ;enter read/write dispatcher

                        ;;      Readhs - Read Host Sector.
                        ;
01DA'   3E 01           readhs: ld      a,1             ;set read operation
                                skip    $+2             ;jump over write entry point
01DC'   26          +           db      26h

                        ;;      Wriths - Write Host Sector.
                        ;
01DD'   AF              wriths: xor     a               ;set write operation
01DE'   21 0213"                ld      hl,hstcmd

                        ;;      Rdwrhs - Read or Write Host Sector.
                        ;
                        ;       Entry:  HL = Physical command request address
                        ;                A = 0 to write
                        ;                A = 1 to read
                        ;
                        ;       Exit:   A  =  0, if no errors
                        ;               A  = -1, if errors
                        ;               Z  = condition of A reg
                        ;
01E1'   77              rdwrhs: ld      (hl),a          ;set driver operation
01E2'   CD 02DF'                call    xdr             ;execute driver read or write
01E5'   21 02BE'                ld      hl,erflag       ;merge error flag for directory protection
01E8'   B6                      or      (hl)
01E9'   77                      ld      (hl),a
01EA'   C9                      ret

                        ;;      Read - Read CP/M Sector.
                        ;
                        ;       Entry:  Seldsk, Settrk, Setsec, Setdma previously called
                        ;
                        ;       Exit:   A =  0 if no errors
                        ;               A = -1 if errors
                        ;
01EB'   AF              read:   xor     a               ;clear unalloc processing
01EC'   32 0226'                ld      (unacnt),a
01EF'   0E 00                   ld      c,wrall         ;inhibit buffer flush after read
01F1'   3C                      inc     a               ;set read operation
                                skip    $+2
```

```
01F2'   26          +           db      26h

                        ;;      Write - Write CP/M Sector.
                        ;
                        ;       Entry:  Seldsk, Settrk, Setsec, Setdma previously called
                        ;
                        ;       Exit:   A =  0 if no errors
                        ;               A = -1 if errors
                        ;               C = Write type
                        ;
01F3'   AF              write:  xor     a               ;set write operation

                        ;;      Rdwr - Read or Write.
                        ;
                        ;       Entry:   A = 0 to write
                        ;                A = 1 to read
                        ;
01F4'   32 02AE'        rdwr:   ld      (readop),a      ;set read/write switch
01F7'   AF                      xor     a               ;reset error flag
01F8'   32 02BE'                ld      (erflag),a
01FB'   2A 020C"                ld      hl,(seksec)     ;set seek host sector
01FE'   22 0208"                ld      (sekhst),hl
0201'   3A 029B'                ld      a,(secmsk)      ;set sector size
0204'   B7                      or      a
0205'   28 CB                   jr      z,rdwrs         ;if deblocking not required
0207'   F6 00                   or      0               ;check track zero single density flag
0208'                   trkzfl  equ     $-1
0209'   F2 0212'                jp      p,rdwr1         ;if track zero not single density
020C'   3A 0206"                ld      a,(sektrk)      ;set seek track
020F'   B7                      or      a
0210'   28 C0                   jr      z,rdwrs         ;if track 0, read or write without deblocking
0212'   79              rdwr1:  ld      a,c             ;save write type
0213'   32 02BA'                ld      (wrtype),a
0216'   FE 02                   cp      wrual
0218'   20 0B                   jr      nz,writ1        ;if not write to unallocated group
021A'   3E 00                   ld      a,0             ;set records per block
021B'                   rpb     equ     $-1
021C'   32 0226'                ld      (unacnt),a      ;start counting unallocated writes
021F'   11 020E"                ld      de,unadsk       ;set unallocated parameter block address
0222'   CD 02D6'                call    cpb             ;copy parameter block
0225'   3E 00           writ1:  ld      a,0             ;set remaining unallocated sectors
0226'                   unacnt  equ     $-1
0227'   B7                      or      a
0228'   28 2E                   jr      z,writ4         ;if not processing unallocated group
022A'   3D                      dec     a
022B'   32 0226'                ld      (unacnt),a      ;update unallocated sectors remaining
022E'   21 0205"                ld      hl,sekdsk       ;set seek parameters
0231'   11 020E"                ld      de,unadsk       ;set unallocated parameters
0234'   CD 02CC'                call    cmp             ;compare parameter blocks
0237'   20 1B                   jr      nz,writ3        ;if not seek to unallocated sector
0239'   2A 0211"                ld      hl,(unasec)     ;advance unallocated sector
023C'   23                      inc     hl
023D'   22 0211"                ld      (unasec),hl
0240'   11 0000                 ld      de,0            ;set sectors per track
0241'                   dpbadr  equ     $-2
0243'   ED 52                   sbc     hl,de
```

```
0245'   20 0A                     jr      nz,writ2        ;if not end of track
0247'   22 0211"                  ld      (unasec),hl     ;reset to sector zero
024A'   2A 020F"                  ld      hl,(unatrk)     ;advance unallocated track
024D'   23                        inc     hl
024E'   22 020F"                  ld      (unatrk),hl
0251'   AF                writ2:  xor     a               ;mark pre-read not required
0252'   18 05                     jr      rwoper
0254'   AF                writ3:  xor     a               ;clear unallocated processing
0255'   32 0226'                  ld      (unacnt),a
0258'   3C                writ4:  inc     a               ;mark pre-read required

                          ;;      Rwoper - Read or Write Operation Proper.
                          ;
0259'   32 028B'          rwoper: ld      (rsflag),a      ;set pre-read block flag
025C'   3A 029B'                  ld      a,(secmsk)      ;set shift counter
025F'   2A 020C"                  ld      hl,(seksec)
0262'   CB 3C             rwop1:  srl     h               ;compute host sector = cpmsec/(2**sekmsk)
0264'   CB 1D                     rr      l
0266'   CB 3F                     srl     a
0268'   20 F8                     jr      nz,rwop1        ;if shift incomplete
026A'   22 0208"                  ld      (sekhst),hl     ;set seek host sector
026D'   F6 00                     or      0               ;check host active flag
026E'                     hstact  equ     $-1
026F'   3E 01                     ld      a,1
0271'   32 026E'                  ld      (hstact),a      ;host buffer always becoms active
0274'   28 0E                     jr      z,rwop2         ;if host buffer was not active
0276'   21 0215"                  ld      hl,hstdsk       ;set active host buffer identification
0279'   11 0205"                  ld      de,sekdsk       ;set seek identification
027C'   CD 02CC'                  call    cmp             ;compare seek request with active host sector
027F'   28 16                     jr      z,rwop3         ;if host buffer contains seek sector
0281'   CD 02C2'                  call    flush           ;flush buffer if previously written
0284'   11 0215"          rwop2:  ld      de,hstdsk       ;set host request block address
0287'   CD 02D6'                  call    cpb             ;copy seek parameter block to host
028A'   3E 00                     ld      a,0             ;check pre-read required
028B'                     rsflag  equ     $-1
028C'   B7                        or      a
028D'   C4 01DA'                  call    nz,readhs       ;read host sector if preread required
0290'   B7                        or      a
0291'   C4 014F'                  call    nz,clract       ;clear host buffer active if read errors
0294'   32 02C3'                  ld      (hstwrt),a      ;mark buffer not written into
0297'   3A 020C"          rwop3:  ld      a,(seksec)      ;set seek sector
029A'   E6 00                     and     0               ;form host buffer index from sector mask
029B'                     secmsk  equ     $-1
029C'   1F                        rra
029D'   57                        ld      d,a             ;multiply index by 128 bytes/sector
029E'   3E 00                     ld      a,0
02A0'   1F                        rra
02A1'   5F                        ld      e,a
02A2'   2A 021A"                  ld      hl,(hstdma)     ;set host buffer address
02A5'   19                        add     hl,de           ;form seek buffer address
02A6'   ED 5B 020A"               ld      de,(sekdma)     ;set user transfer address
02AA'   01 0080                   ld      bc,128          ;set CP/M sector length
02AD'   3E 00                     ld      a,0             ;set transfer direction
02AE'                     readop  equ     $-1
02AF'   B7                        or      a
02B0'   20 05                     jr      nz,rwop4        ;if read operation
```

```
02B2'   EB                      ex      de,hl           ;switch directions
02B3'   3C                      inc     a
02B4'   32 02C3'                ld      (hstwrt),a      ;mark buffer written into
02B7'   ED B0           rwop4:  ldir                    ;move sector to/from user buffer
02B9'   3E 00                   ld      a,0             ;set write type
02BA'                   wrtype  equ     $-1
02BB'   FE 01                   cp      wrdir
02BD'   3E 00                   ld      a,0             ;set error flag
02BE'                   erflag  equ     $-1
02BF'   C0                      ret     nz
02C0'   B7                      or      a
02C1'   C0                      ret     nz              ;if errors, do not clobber directory

                        ;;      Flush - Flush buffer to disk.
                        ;
02C2'   3E 00           flush:  ld      a,0             ;check host written flag
02C3'                   hstwrt  equ     $-1
02C4'   B7                      or      a
02C5'   C4 01DD'                call    nz,wriths       ;if buffer written into, write host sector
02C8'   32 02C3'                ld      (hstwrt),a      ;clear host written flag if no errors
02CB'   C9                      ret

                        ;;      Cmp - Compare Paramater Blocks.
                        ;
                        ;       Entry:  HL = Parameter block
                        ;               DL = Parameter block
                        ;
                        ;       Exit:   Z  = Set   if parameters identical
                        ;               Z  = Clear if parameters different
                        ;
02CC'   06 05           cmp:    ld      b,5             ;set length of parameter block
02CE'   1A              cmp1:   ld      a,(de)          ;compare next byte
02CF'   AE                      xor     (hl)
02D0'   C0                      ret     nz              ;if parameters different
02D1'   13                      inc     de
02D2'   23                      inc     hl
02D3'   10 F9                   djnz    cmp1            ;if more bytes
02D5'   C9                      ret

                        ;;      Cpb - Copy Parameter Block.
                        ;
                        ;       Entry:  DE = Address of Unallocated or Host parameter block
                        ;
                        ;       Exit:   Seek parameter block copied into block at DE
                        ;
02D6'   21 0205"        cpb:    ld      hl,sekdsk       ;set source parameters
02D9'   01 0005                 ld      bc,5            ;set block length
02DC'   ED B0                   ldir                    ;copy parameter block
02DE'   C9                      ret

                        ;;      Xdr - Execute Driver Request.
                        ;
                        ;       Entry:  HL = pointer to Physical Driver Request Block
                        ;
                        ;       Exit:   Physical Driver exit condition are maintained if
                        ;               no errors or user did not request warm start.
```

```
                                   ;
        02DF'   22 02EA'           xdr:    ld      (xdra),hl       ;save request for retrys
        02E2'   2A 02EA'           xdr1:   ld      hl,(xdra)       ;restore request address
        02E5'   CD F02A                    call    Xqdvr           ;execute physical driver
        02E8'   47                         ld      b,a             ;save read/write error status
        02E9'   3A 02EA'                   ld      a,(xdra)        ;get driver operation
        02EA'                      xdra    equ     $-2
        02EC'   4F                         ld      c,a             ;set message index
        02ED'   3C                         inc     a
        02EE'   20 05                      jr      nz,xdr2         ;if not select request
        02F0'   B5                         or      l
        02F1'   B4                         or      h
        02F2'   C0                         ret     nz              ;if dph address returned by driver
        02F3'   18 03                      jr      xdr3
        02F5'   78                 xdr2:   ld      a,b             ;set read/write error status
        02F6'   B7                         or      a
        02F7'   C8                         ret     z               ;if no read/write errors
        02F8'   2A 02EA'           xdr3:   ld      hl,(xdra)       ;put drive name in message
        02FB'   23                         inc     hl
        02FC'   23                         inc     hl
        02FD'   7E                         ld      a,(hl)
        02FE'   C6 41                      add     a,'A'
        0300'   32 0330'                   ld      (xdrb),a
        0303'   0C                         inc     c
        0304'   20 0C                      jr      nz,xdr4         ;if not select request
        0306'   CD 0375'                   call    pmsgi
        0309'   53 65 6C 65                db      'Select',0
        030D'   63 74 00
        0310'   18 16                      jr      xdr6
        0312'   0D                 xdr4:   dec     c
        0313'   20 0B                      jr      nz,xdr5         ;if not write request
        0315'   CD 0375'                   call    pmsgi
        0318'   57 72 69 74                db      'Write',0
        031C'   65 00
        031E'   18 08                      jr      xdr6
        0320'   CD 0375'           xdr5:   call    pmsgi           ;must be read request
        0323'   52 65 61 64                db      'Read',0
        0327'   00
        0328'   CD 0115'           xdr6:   call    pmsg
        032B'   20 45 72 72                db      ' Err '
        032F'   20
        0330'   64 3A 20           xdrb:   db      'd: '
        0333'   41 28 63 63                db      'A(ccept), '
        0337'   65 70 74 29
        033B'   2C 20
        033D'   49 28 67 6E                db      'I(gnore), '
        0341'   6F 72 65 29
        0345'   2C 20
        0347'   52 28 65 74                db      'R(etry) '
        034B'   72 79 29 20
        034F'   00                         db      0
        0350'   CD 0009'                   call    bconin          ;read character from console
        0353'   F5                         push    af
        0354'   CD 0115'                   call    pmsg
        0357'   0D                         db      cr
        0358'   521B                       dw      dellin
```

```
035A'   0B 00                   db      up,0
035C'   67                      ld      h,a             ;zero dph for accepted or ignored select errors
035D'   6F                      ld      l,a
035E'   F1                      pop     af
035F'   E6 5F                   and     5fh             ;ignore parity, case
0361'   FE 03                   cp      3
0363'   28 0A                   jr      z,xdr7          ;if warm start requested
0365'   D6 49                   sub     'I'
0367'   C8                      ret     z               ;if user ignored error, don't tell BDOS
0368'   D6 F8                   sub     'A'-'I'
036A'   C2 02E2'                jp      nz,xdr1         ;retry request
036D'   2F                      cpl
036E'   C9                      ret

036F'   CD 01A5'        xdr7:   call    cad             ;clear active disk
0372'   C3 0003'                jp      bwboot

0375'   CD 0115'        pmsgi:  call    pmsg
0378'   0D 0A                   db      cr,lf
037A'   451B                    dw      inslin
037C'   00                      db      0
037D'   C3 0115'                jp      pmsg

                                subttl  Deblocker Storage Area
                                page
```

```
XEROX 820-II BIOS       MACRO-80 3.44   09-Dec-81
Deblocker Storage Area

                                reserve macro   s,n
                                s       equ     $+.
                                .       aset    .+n
                                        endm
         0000                   .       aset    0
         0380'                          dseg

                                ;;      Host Sector Deblocking Buffer.
                                ;
                                reserve hstbuf,512

                                ;;      Physical Driver Select Command.
                                ;
                                reserve selcmd,3                ;select command, unit, drive

                                ;;      Seek Sector Parameter Block.
                                ;
                                reserve sekcmd,1                ;kindly
                                reserve sekunt,1                ; leave
                                reserve sekdsk,1                ;  these
                                reserve sektrk,2                ;   bytes
                                reserve sekhst,2                ;    alone
                                reserve sekdma,2
                                reserve seksec,2

                                ;;      Unallocated Sector Parameter Block.
                                ;
                                reserve unadsk,1                ;kindly
                                reserve unatrk,2                ; leave
                                reserve unasec,2                ;  these

                                ;;      Host Sector Parameter Block.
                                ;
                                reserve hstcmd,1                ;kindly
                                reserve hstunt,1                ; leave
                                reserve hstdsk,1                ;  these
                                reserve hsttrk,2                ;   bytes
                                reserve hstsec,2                ;    alone
                                reserve hstdma,2

                                ;;      Disk Parameter Header Addresses.
                                ;
                                reserve dphtab,('P'-'A'+1)*2

         0000"                          cseg

                                        end
```

```
XEROX 820-II BIOS       MACRO-80 3.44   09-Dec-81
Deblocker Storage Area

Macros:
RESERVE         SKIP

Symbols:
023C    .               0009'   BCONIN          000C'   BCONOT
0006'   BCONST          0E00    BDOSLN          0018'   BHOME
0000'   BIOS            000F'   BPRINT          002D'   BPRNTS
0012'   BPUNCH          0027'   BREAD           0015'   BREADR
0030'   BSCTRN          001B'   BSELD           0024'   BSETD
0021'   BSETS           001E'   BSETT           0003'   BWBOOT
002A'   BWRIT           01A5'   CAD             01B4'   CAD1
00F1'   CBOOT           0800    CCPLEN          0004    CDISK
014F'   CLRACT          02CC'   CMP             02CE'   CMP1
F04E    CONIN           F051    CONOUT          F04B    CONST
02D6'   CPB             000D    CR              013F'   DBOOT
014B'   DBT2            521B    DELLIN          013B'   DMABAS
0241'   DPBADR          021C"   DPHTAB          02BE'   ERFLAG
001B    ESC             02C2'   FLUSH           01B6'   HOME
026E'   HSTACT          0000"   HSTBUF          0213"   HSTCMD
021A"   HSTDMA          0215"   HSTDSK          0218"   HSTSEC
0216"   HSTTRK          0214"   HSTUNT          02C3'   HSTWRT
0033'   INITIO          451B    INSLIN          000A    LF
F054    LIST            F057    LISTST          0121'   MLS
0130'   MLS1            0132'   MLS2            F000    MONITR
002C    NSECTS          0115'   PMSG            0375'   PMSGI
F060    PUNCH           01F4'   RDWR            0212'   RDWR1
01E1'   RDWRHS          01D2'   RDWRS           01EB'   READ
F05D    READER          01DA'   READHS          02AE'   READOP
0062    REV             021B'   RPB             028B'   RSFLAG
0262'   RWOP1           0284'   RWOP2           0297'   RWOP3
02B7'   RWOP4           0259'   RWOPER          00D0'   SECCNT
029B'   SECMSK          01C8'   SECTRN          0203"   SEKCMD
020A"   SEKDMA          0205"   SEKDSK          0208"   SEKHST
020C"   SEKSEC          0206"   SEKTRK          0204"   SEKUNT
015A'   SEL1            016D'   SEL2            0183'   SEL3
01A5'   SEL4            0200"   SELCMD          0154'   SELDSK
01C3'   SETDMA          01BE'   SETSEC          01B9'   SETTRK
00DC'   SPT             0133'   TRANSZ          0208'   TRKZFL
0226'   UNACNT          020E"   UNADSK          0211"   UNASEC
020F"   UNATRK          000B    UP              0069'   WBOOT
00B0'   WBT1            00B6'   WBT2            00D8'   WBT3
00E6'   WBT4            0034'   WBT5            0050'   WBTCOM
0058'   WBTERR          0000    WRALL           0001    WRDIR
0225'   WRIT1           0251'   WRIT2           0254'   WRIT3
0258'   WRIT4           01F3'   WRITE           01DD'   WRITHS
02BA'   WRTYPE          0002    WRUAL           02DF'   XDR
02E2'   XDR1            02F5'   XDR2            02F8'   XDR3
0312'   XDR4            0320'   XDR5            0328'   XDR6
036F'   XDR7            02EA'   XDRA            0330'   XDRB
0122'   XLATE           F02A    XQDVR


No Fatal error(s)
```

	689#	695	695#	695	699	699#	699	703	703#	703	704	704#
	704	705	705#	705	706	706#	706	707	707#	707	708	708#
	708	709	709#	709	713	713#	713	714	714#	714	715	715#
	715	719	719#	719	720	720#	720	721	721#	721	722	722#
	722	723	723#	723	724	724#	724	728	728#	728		
BCONIN	14#	65	654									
BCONOT	15#	167										
BCONST	13#											
BDOSLN	37#	40	46	55	71							
BHOME	20#											
BIOS	10#	46	55	71	115							
BPRINT	16#											
BPRNTS	28#											
BPUNCH	17#											
BREAD	25#											
BREADR	18#											
BSCTRN	29#											
BSELD	21#	73										
BSETD	24#	118										
BSETS	23#	113										
BSETT	22#	108										
BWBOOT	11#	52	674									
BWRIT	26#											
CAD	333#	673										
CAD1	337	341#										
CBOOT	10	146#										
CCPLEN	38#	40	55	71								
CDISK	39#	54	231#	333								
CLRACT	266#	529										
CMP	483	519	577#									
CMP1	578#	583										
CONIN	14	203#										
CONOUT	15	204#										
CONST	13	202#										
CPB	474	523	592#									
CR	222#	657	677									
DBOOT	67	256#										
DBT2	260#	262										
DELLIN	226#	658										
DMABAS	72	134	138	192#								
DPBADR	315	489#										
DPHTAB	258	279	728#									
ERFLAG	418	454	555#									
ESC	223#	225	226									
FLUSH	521	562#										
HOME	20	346#										
HSTACT	267	513#	515									
HSTBUF	256	695#										

HSTCMD	404	719#										
HSTDMA	257	539	724#									
HSTDSK	517	522	721#									
HSTSEC	723#											
HSTTRK	722#											
HSTUNT	720#											
HSTWRT	530	549	563#	566								
INITIO	31#											
INSLIN	225#	678										
LF	220#	677										
LIST	16	205#										
LISTST	28	206#										
MLS	82	111	172#									
MLS1	180	183#										
MLS2	176	185#										
MONITR	200#	202	203	204	205	206	207	208				
NSECTS	40#	69										
PMSG	61	146	160#	168	641	656	676	680				
PMSGI	628	634	638	676#								
PUNCH	17	208#										
RDWR	452#											
RDWR1	462	466#										
RDWRHS	393	416#										
RDWRS	391#	459	465									
READ	25	119	430#									
READER	18	207#										
READHS	397#	527										
READOP	391	452	544#									
RESERVE	685#	694	698	702	703	704	705	706	707	708	712	713
	714	718	719	720	721	722	723	727				
REV	41#	152										
RPB	320	471#										
RSFLAG	504	525#										
RWOP1	507#	510										
RWOP2	516	522#										
RWOP3	520	531#										
RWOP4	546	550#										
RWOPER	497	504#										
SECCNT	70	123#	125									
SECMSK	326	457	505	533#								
SECTRN	29	378#										
SEKCMD	392	703#										
SEKDMA	366	541	708#									
SEKDSK	277	334	481	518	592	705#						
SEKHST	456	511	707#									
SEKSEC	359	455	506	531	709#							
SEKTRK	352	463	706#									
SEKUNT	704#											
SEL1	279#	291										

SEL2	283	292#			
SEL3	289	308#			
SEL4	307	329#			
SELCMD	293	699#			
SELDSK	21	276#			
SETDMA	24	366#			
SETSEC	23	359#			
SETTRK	22	352#			
SKIP	243#	398	434		
SPT	92	131#	178		
TRANSZ	84	186#			
TRKZFL	324	461#			
UNACNT	431	472	476#	480	499
UNADSK	473	482	713#		
UNASEC	485	487	492	715#	
UNATRK	493	495	714#		
UP	221#	659			
WBOOT	11	66#			
WBT1	101	104	106#	142	
WBT2	110#	132			
WBT3	117	127#			
WBT4	136#	137			
WBT5	45#	126			
WBTCOM	56#	58			
WBTERR	48	61#	76	121	
WRALL	235#	432			
WRDIR	236#	553			
WRIT1	469	475#			
WRIT2	491	496#			
WRIT3	484	498#			
WRIT4	478	500#			
WRITE	26	445#			
WRITHS	403#	565			
WRTYPE	467	552#			
WRUAL	237#	468			
XDR	298	417	604#		
XDR1	605#	669			
XDR2	612	617#			
XDR3	616	620#			
XDR4	627	632#			
XDR5	633	638#			
XDR6	631	637	641#		
XDR7	665	673#			
XDRA	604	605	608	609#	620
XDRB	625	644#			
XLATE	79	173#			
XQDVR	230#	606			

Notes

```
        Title   Banked Physical Driver

;;      Banked Physical Driver
;
;       Copyright (C) 1982, Balcones Computer Corporation
;
;       Transferred to Public Domain - (PD) 1983
;
;;      After executing this program by entering BANK x:(where x is any valid
;       CP/M disk drive A-P).  The BANK program will load a physical disk
;       driver into memory.  This physical driver is executed when drive x: is
;       accessed by CP/M.  This particular disk driver will map normal CP/M
;       files onto the address space of the alternate memory bank
;       (bank 0) in the 820-II.
;
;       This utility demonstrates the flexibility of the logical to
;       physical disk mapping in the 820-II.  The BANK program
;       moves the physical disk driver up to high memory.
;       It then updates the entry for drive x: in the logical to physical
;       disk drive mapping table telling the system to use physical
;       disk driver #3 when CP/M requests service from drive x:.
;
;       The execution address of the BANK driver is then placed in
;       entry #3 of the physical disk driver address table.
;
;       If BANK is executed by entering:  A>BANK P:
;
;       Then doing a A>DIR P: would display the following directory:
;
;       BOOT    .ROM  :  OPTION  .ROM  :  SCREEN  .MEM  :  EXPAND  .RAM
;
;       Entering: A>STAT P:*.* will display the following:
;
;                        Recs  Bytes  Ext Acc
;                          64    12k    1 R/O P:BOOT.ROM
;                         256    32k    1 R/W P:EXPAND.RAM
;                          16     2k    1 R/W P:OPTION.ROM
;                          24     4k    1 R/W P:SCREEN.MEM
;                       Bytes Remaining On P: 0k
;
;       The files map to the following memory addresses in bank 0:
;
;                       BOOT.ROM        0000h-2fffh
;                       EXPAND.RAM      4000h-bfffh
;                       OPTION.ROM      17ffh-1fffh
;                       SCREEN.MEM      3000h-3bffh
;
;       The BANK program can also be a very useful tool in that after
;       it has been executed a high level language program can access
;       items in the alternate memory bank as disk files on drive x:
;
;       Of particular interest is the file SCREEN.MEM, notice that it
;       is 24 records long.  Each record (128 bytes) corresponds to a
;       line on the CRT (only the first 80 bytes of each record are in
;       the display window).  The first record of the file corresponds
;       to the first line of the CRT only if the CRT has not been
```

```
                ;       permitted to scroll since the last clear screen command was sent
                ;       to it.
                ;
                        Subttl  Constants & Program Mover
                        page
```

```
 F000                 Monitr  equ     0f000h          ;Base address of resident monitor
 F033                 Xcrtmv  equ     monitr+33h      ;Crt <-> Ram Move LDIR Simulator
 F036                 Xgetsl  equ     monitr+36h      ;Get driver select table address to hl

 FF3C                 Bavail  equ     0ff3ch          ;Pointer to beginning of available memory
 FF3E                 Eavail  equ     0ff3eh          ;Pointer to end of available memory

 0005                 bdos    equ     5
 005C                 dfcb    equ     5ch

 FA80                 drvadr  equ     0fa80h          ;address for Bank driver
 0000                 stack   equ     0

                              .z80

 0000'                        Aseg
                              Org     100h
 0100    18 5A                jr      loadit

 0102    43 6F 70 79          db      'Copyright (C) 1982 Balcones Computer Corporation'
 0106    72 69 67 68
 010A    74 20 28 43
 010E    29 20 31 39
 0112    38 32 20 42
 0116    61 6C 63 6F
 011A    6E 65 73 20
 011E    43 6F 6D 70
 0122    75 74 65 72
 0126    20 43 6F 72
 012A    70 6F 72 61
 012E    74 69 6F 6E
 0132    20 54 72 61          db      'Transferred to Public Domain - (PD) 1983',26
 0136    6E 73 66 65
 013A    72 72 65 64
 013E    20 74 6F 20
 0142    50 75 62 6C
 0146    69 63 20 44
 014A    6F 6D 61 69
 014E    6E 20 2D 20
 0152    28 50 44 29
 0156    20 31 39 38
 015A    33 1A


 015C    CD 03D8      loadit: call    req822          ;see if machine is 820-II
 015F    3A 005C              ld      a,(dfcb)
 0162    B7                   or      a
 0163    28 30                jr      z,bnkusg
 0165    F5                   push    af
 0166    CD 03F8              call    ckspac          ;see if room for driver
 0169    F1                   pop     af
 016A    3D                   dec     a
 016B    4F                   ld      c,a
```

```
016C   21 0259                   ld      hl,driver
016F   11 FA80                   ld      de,banked
0172   D5                        push    de
0173   C5                        push    bc
0174   01 017F                   ld      bc,drvlen
0177   ED B0                     ldir
0179   26 00                     ld      h,0             ;indicate register return
017B   CD F036                   call    XGetsl          ;get select table address
017E   C1                        pop     bc
017F   E5                        push    hl
0180   09                        add     hl,bc
0181   09                        add     hl,bc
0182   36 03                     ld      (hl),3
0184   23                        inc     hl
0185   36 00                     ld      (hl),0
0187   E1                        pop     hl
0188   11 0026                   ld      de,2*16+3*2
018B   19                        add     hl,de
018C   D1                        pop     de
018D   73                        ld      (hl),e
018E   23                        inc     hl
018F   72                        ld      (hl),d
0190   0E 0D                     ld      c,13
0192   C3 0005                   jp      bdos

0195   11 019D          bnkusg:  ld      de,bnkmsg
0198   0E 09                     ld      c,9
019A   C3 0005                   jp      bdos

019D   55 73 61 67      bnkmsg:  db      'Usage: BANK x:$'
01A1   65 3A 20 42
01A5   41 4E 4B 20
01A9   78 3A 24

01AC                             ds      200h-103h-($-loadit),-1
0259                    driver:
                                 .phase  Drvadr

                                 Subttl  Bank Driver
                                 page
```

```
FA80    7E              banked: ld      a,(hl)          ;get driver op
FA81    4F                      ld      c,a
FA82    23                      inc     hl
FA83    3C                      inc     a
FA84    28 51                   jr      z,selbnk        ;if select op
FA86    23                      inc     hl
FA87    23                      inc     hl
FA88    56                      ld      d,(hl)          ;set track
FA89    23                      inc     hl
FA8A    23                      inc     hl
FA8B    7E                      ld      a,(hl)          ;set sector
FA8C    0F                      rrca
FA8D    5F                      ld      e,a
FA8E    23                      inc     hl
FA8F    23                      inc     hl
FA90    7E                      ld      a,(hl)          ;set transfer address
FA91    23                      inc     hl
FA92    66                      ld      h,(hl)
FA93    6F                      ld      l,a
FA94    06 00                   ld      b,0             ;preset crtldir op
FA96    7C                      ld      a,h
FA97    FE C0                   cp      0c0h
FA99    30 01                   jr      nc,bank1        ;if transfer outside banked area
FA9B    05                      dec     b               ;set ram->crt
FA9C    79              bank1:  ld      a,c             ;set read/write op
FA9D    B7                      or      a
FA9E    28 02                   jr      z,bank2         ;if write
FAA0    06 01                   ld      b,1             ;set crt->ram
FAA2    C5              bank2:  push    bc              ;save direction op
FAA3    B2                      or      d               ;check directory track
FAA4    FA FAC8                 jp      m,bank6         ;if directory operation
FAA7    FE 30                   cp      030h
FAA9    79                      ld      a,c             ;set read/write switch
FAAA    01 0080                 ld      bc,128
FAAD    38 05                   jr      c,bank3         ;if not within screen memory
FAAF    B7                      or      a
FAB0    20 05                   jr      nz,bank4        ;if read
FAB2    0E 50                   ld      c,80            ;only write one line
FAB4    B7              bank3:  or      a               ;test read/write
FAB5    28 01                   jr      z,bank5         ;if write
FAB7    EB              bank4:  ex      de,hl           ;set read
FAB8    F1              bank5:  pop     af              ;get mover op to A
FAB9    ED 73 FAC4              ld      (stksav),sp     ;use high stack
FABD    31 0000                 ld      sp,stack
FAC0    CD F033                 call    Xcrtmv          ;move it to/from crt bank
FAC3    31 0000                 ld      sp,0
FAC4                    stksav  equ     $-2
FAC6    AF                      xor     a               ;always succeeds
FAC7    C9                      ret
FAC8    11 FADB         bank6:  ld      de,Direct       ;set directory address
FACB    0D                      dec     c
FACC    20 01                   jr      nz,bank7        ;if directory write
FACE    EB                      ex      de,hl
```

```
        FACF    01 0080         bank7:  ld      bc,80h
        FAD2    ED B0                   ldir                    ;read or re-write directory
        FAD4    F1                      pop     af
        FAD5    AF                      xor     a
        FAD6    C9                      ret

        FAD7    21 FB58         selbnk: ld      hl,dph
        FADA    C9                      ret

                                        Subttl  Directory Sector, Dpb & Dph
                                        page
```

```
FADB    00              Direct: db      0
FADC    42 4F 4F 54             dc      'BOOT    R'
FAE0    20 20 20 20
FAE4    D2
FAE5    4F 4D                   db      'OM'
FAE7    00 00 00 40             db      00,00,00,64
FAEB    01 02 03 04             db      01,02,03,04     ;Bank 0 Memory locations        0000h-1fffh
FAEF    05 06 00 00             db      05,06,00,00     ;                               2000h-2fffh
FAF3    00 00 00 00             db      00,00,00,00
FAF7    00 00 00 00             db      00,00,00,00
FAFB    00                      db      0
FAFC    4F 50 54 49             db      'OPTION  ROM'
FB00    4F 4E 20 20
FB04    52 4F 4D
FB07    00 00 00 10             db      00,00,00,16     ;Bank 0 Memory locations        17ffh-1fffh
FB0B    04 00 00 00             db      04,00,00,00
FB0F    00 00 00 00             db      00,00,00,00
FB13    00 00 00 00             db      00,00,00,00
FB17    00 00 00 00             db      00,00,00,00
FB1B    00                      db      0
FB1C    53 43 52 45             db      'SCREEN  MEM'
FB20    45 4E 20 20
FB24    4D 45 4D
FB27    00 00 00 18             db      0,0,0,24
FB2B    07 08 00 00             db      07,08,00,00     ;Bank 0, Memory locations       3000h-3bffh
FB2F    00 00 00 00             db      00,00,00,00
FB33    00 00 00 00             db      00,00,00,00
FB37    00 00 00 00             db      00,00,00,00
FB3B    00                      db      0
FB3C    45 58 50 41             db      'EXPAND  RAM'
FB40    4E 44 20 20
FB44    52 41 4D
FB47    01 00 00 80             db      01,00,00,80h
FB4B    09 0A 0B 0C             db      09,10,11,12     ;Bank 0, Memory locations       4000h-5fffh
FB4F    0D 0E 0F 10             db      13,14,15,16     ;                               6000h-7fffh
FB53    11 12 13 14             db      17,18,19,20     ;                               8000h-9fffh
FB57    15 16 17 18             db      21,22,23,24     ;                               a000h-bfffh
FB5B    0000 0000       dph:    dw      0,0,0,0
FB5F    0000 0000
FB63    FB7F FB6B               dw      dirbuf,dpb
FB67    0000 FB7B               dw      0,alloc
FB6B    0002            dpb:    dw      2               ;spt
FB6D    04 0F 01                db      4,15,1          ;blkshf, blkmsk, nullmsk
FB70    0018 0003               dw      24,3,128,0,-8   ;dsw,dirm,alloc01,chksiz,trk off
FB74    0080 0000
FB78    FFF8
FB7A    00                      db      0               ;128 byte sectors
```

```
Banked Physical Driver                          MACRO-80 3.44   09-Dec-81
Directory Sector, Dpb & Dph

      FB7B                    alloc:  ds      4               ;allocation vector
      FB7F                    dirbuf: ds      128             ;directory buffer

                                      .dephase
      017F                    drvlen  equ     $-driver

                                      Subttl  System Identification
                                      page
```

```
                               ;;      Verify The machine this program is being run by Murphy or
                               ;       a Xerox 820-II.
                               ;
03D8    3A F000                Req822: ld      a,(monitr)      ;make certain system is an 820-II
03DB    FE C3                          cp      0c3h            ;should be a jump instruction if 820
03DD    20 0D                          jr      nz,notii        ;if not give error message
03DF    2A F001                        ld      hl,(monitr+1)   ;follow reload monitor jump
03E2    7E                             ld      a,(hl)
03E3    FE F3                          cp      0f3h
03E5    20 05                          jr      nz,notii        ;if interrupts not disabled
03E7    23                             inc     hl
03E8    7E                             ld      a,(hl)
03E9    FE DB                          cp      0dbh
03EB    C8                             ret     z
03EC    E1                     Notii:  pop     hl              ;pitch return address
03ED    11 0434                        ld      de,msg
03F0    0E 09                  pmsg:   ld      c,9
03F2    CD 0005                        call    bdos
03F5    C3 0000                        jp      0

                               ;;      The pointer at Bavail points to the start of free memory, Eavail
                               ;       points to the end of free memory.  This test verifies that there
                               ;       is enough space for this program to fit in this un-allocated memory
                               ;       space.  If so the Eavail pointer is updated to the start of the driver -1.
                               ;       If not an error message is sent to the console.
                               ;
03F8    ED 5B FF3C             Ckspac: ld      de,(bavail)     ;get pointer to start of free address space
03FC    21 FA80                        ld      hl,drvadr       ;start of driver
03FF    B7                             or      a
0400    ED 52                          sbc     hl,de
0402    38 11                          jr      c,nroom         ;if drvadr < bavail then no space
0404    2A FF3E                        ld      hl,(eavail)     ;get pointer to end of available space
0407    11 FBFF                        ld      de,drvadr+drvlen
040A    ED 52                          sbc     hl,de
040C    38 07                          jr      c,nroom         ;if driver end > end of eavail then no space
040E    21 FA80                        ld      hl,drvadr       ;else update end pointer
0411    22 FF3E                        ld      (eavail),hl
0414    C9                             ret
0415    11 041A                nroom:  ld      de,nspace
0418    18 D6                          jr      pmsg

041A    46 72 65 65            Nspace: db      'Free memory space in use.$'
041E    20 6D 65 6D
0422    6F 72 79 20
0426    73 70 61 63
042A    65 20 69 6E
042E    20 75 73 65
0432    2E 24

0434    54 68 69 73            Msg:    db      'This program requires a Xerox 820-II Information Processor.$'
0438    20 70 72 6F
043C    67 72 61 6D
0440    20 72 65 71
```

```
0444      75 69 72 65
0448      73 20 61 20
044C      58 65 72 6F
0450      78 20 38 32
0454      30 2D 49 49
0458      20 49 6E 66
045C      6F 72 6D 61
0460      74 69 6F 6E
0464      20 50 72 6F
0468      63 65 73 73
046C      6F 72 2E 24

                                          Subttl  Symbol Table
                                          end
```

```
Banked Physical Driver                          MACRO-80 3.44   09-Dec-81
Symbol Table

Macros:

Symbols:
FB7B    ALLOC           FA9C    BANK1           FAA2    BANK2
FAB4    BANK3           FAB7    BANK4           FAB8    BANK5
FAC8    BANK6           FACF    BANK7           FA80    BANKED
FF3C    BAVAIL          0005    BDOS            019D    BNKMSG
0195    BNKUSG          03F8    CKSPAC          005C    DFCB
FB7F    DIRBUF          FADB    DIRECT          FB6B    DPB
FB5B    DPH             0259    DRIVER          FA80    DRVADR
017F    DRVLEN          FF3E    EAVAIL          015C    LOADIT
F000    MONITR          0434    MSG             03EC    NOTII
0415    NROOM           041A    NSPACE          03F0    PMSG
03D8    REQ822          FAD7    SELBNK          0000    STACK
FAC4    STKSAV          F033    XCRTMV          F036    XGETSL


No Fatal error(s)
```

ALLOC	269	278#			
BANK1	181	183#			
BANK2	185	187#			
BANK3	193	197#			
BANK4	195	199#			
BANK5	198	200#			
BANK6	189	208#			
BANK7	210	212#			
BANKED	118	159#			
BAVAIL	68#	314			
BDOS	71#	140	144	305	
BNKMSG	142	146#			
BNKUSG	111	142#			
CKSPAC	113	314#			
DFCB	72#	109			
DIRBUF	268	279#			
DIRECT	208	225#			
DPB	268	271#			
DPH	218	266#			
DRIVER	117	152#	282		
DRVADR	74#	153	315	320	323
DRVLEN	121	282#	320		
EAVAIL	69#	319	324		
LOADIT	81	108#	151		
MONITR	64#	65	66	291	294
MSG	303	337#			
NOTII	293	297	302#		
NROOM	318	322	326#		
NSPACE	326	329#			
PMSG	304#	327			
REQ822	108	291#			
SELBNK	163	218#			
STACK	75#	202			
STKSAV	201	205#			
XCRTMV	65#	203			
XGETSL	66#	124			

```
                                        Title   Position encoded keyboard handler

                                ;;      Position encoded keyboard handler for the 820-II & 16/8
                                ;       professional computer.
                                ;
                                ;       Copyright 1983 (C) XEROX Corporation
                                ;

                                ;;              This is the stand alone rom addition to the Xerox
                                ;               820-II monitor. It is called once during monitor restart
                                ;               and at that time patches the monitor in ram to
                                ;               call the modified k/b,crt,Screenprint and printer
                                ;               routines.  It then moves in its own SIGNON overlay
                                ;               and jumps into it.
                                ;
                                ;               This SIGNON in addition to selecting the disk driver also
                                ;               moves into ram (in the spare driver area) translation
                                ;               tables and code for k/b and printer routines (crt is run
                                ;               out of rom).
                                ;
                                ;               There is also a RX BOOT overlay which is selected instead
                                ;               of the Xerox one. This loads the national translation
                                ;               tables from disk and then calls the Xerox BOOT.


                                        .z80

        000D                    ver     defl    013

                                        subttl  Xerox ROM dependant equates
                                        page
```

```
                                  ;;      The following equates are dependant on the revision of the Xerox ROM
                                  ;       These are compatible with Ver. 4.02
                                  ;
        1800                      rx1984  equ     1800h           ;start of rx1984
        0800                      romsiz  equ     800h            ;size of eprom
        F000                      monitr  equ     0f000h          ;start of monitor ad jump table
        F1EC                      savstk  equ     x'f1ec'         ;stack save address
        FF10                      ctcvec  equ     x'ff10'         ;counter timer interrupt vector
        FF18                      sysvec  equ     0ff18h          ;vector page
        FF1A                      kbvec   equ     sysvec+2        ;keyboard vector
        FC5D                      tca     equ     0fc5dh          ;start of 4.02 transient command area
        0002                      boff1   equ     ('A'-'@')*2     ;A command vector in command table
        0018                      boff2   equ     ('L'-'@')*2     ;l command vector in command table
        0019                      sioff   equ     19h             ;sioout vector in monitor table
        000A                      kboff   equ     0ah             ;offset in k/b int service for patch
        0010                      fcrtof  equ     10h             ;fast crt out vector in monitor table
        0012                      crtcall equ     12h             ;offset in crt driver for patch
        F006                      const   equ     monitr+6
        F009                      conin   equ     monitr+9
        F003                      warm    equ     monitr+3
        F01B                      select  equ     monitr+1bh
        F01E                      home    equ     monitr+1eh
        F024                      read    equ     monitr+24h
        F03C                      config  equ     monitr+x'3c'    ;monitor configure routine
        F03F                      siordy  equ     monitr+x'3f'    ;sio channel b output ready status
        F066                      idle    equ     monitr+x'66'    ;idle while i/o pending
        F06C                      mntrex  equ     monitr+x'6c'    ;monitor jump table expansion area
        F06C                      kybdlp  equ     monitr+x'6c'    ;low profile keyboard entry address
        F06F                      key2    equ     monitr+x'6f'    ;keyboard xlat char entry address
        F072                      key5    equ     monitr+x'72'    ;keyboard without xlat char entry address
        F075                      pnext   equ     monitr+x'75'    ;print message after call
        F078                      prboff  equ     monitr+x'78'    ;promt boot entry
        0182                      crtd1   equ     0182h
        0196                      crtd2   equ     0196h
        01DD                      grpad   equ     1ddh            ;address of set graphics attribute
        1078                      xrsign  equ     1078h           ;adress of xr signon overlay
        0060                      sigoff  equ     60h             ;offset of ver value in signon
        11C0                      xrboot  equ     11c0h           ;boot o/l address
        F167                      mkey2   equ     x'f167'         ;keyboard handler entry address
        F18F                      mkey5   equ     x'f18f'         ;return from keyboard and timer interrupt add
        F22F                      sprnt1  equ     0f22fh          ;patch address for screen print
        F232                      sprnt2  equ     0f232h          ;return address from RX screenprint code
        F293                      crtoff  equ     x'f293'         ;switch to ram side
        F339                      prvatt  equ     0f339h          ;contains address of current set attribute
        FA62                      prompt  equ     0fa62h          ;4.02 PROMPT
        FA95                      mprmt0  equ     x'fa95'         ;4.01 PRMT0
        FC3D                      mpnext  equ     x'fc3d'         ;4.01 PNEXT

                                  ;       Data Addreses

        ED80                      bootbf  equ     0ed80h
        F0E3                      mask    equ     0f0e3h
        F091                      confg   equ     0f091h
```

```
Position encoded keyboard handler       MACRO-80 3.44   09-Dec-81
Xerox ROM dependant equates

        F20E                        spact   equ     x'f20e'
        F319                        gold    equ     0f319h
        F360                        seltab  equ     0f360h
        F470                        fivdpb  equ     0f470h
        F708                        rigdpb  equ     0f708h
        F800                        tabled  equ     0f800h          ;space for rx code
        FA11                        phytrk  equ     0fa11h
        FF3C                        availb  equ     x'ff3c'         ;bottom available ram memory
        FF50                        intstk  equ     x'ff50'         ;tempory stack address
        FF54                        steprt  equ     0ff54h
        FF5C                        linbuf  equ     0ff5ch
        FFAC                        cursor  equ     0ffach
        FFB2                        leadin  equ     0ffb2h
        FFB3                        attrib  equ     0ffb3h          ;address of attributes enabled flag
        FFB4                        chrsav  equ     0ffb4h

                                    ;       Port addressess
                                    ;
        001D                        sysctl  equ     1dh
        001C                        syspio  equ     1ch
        0005                        siodpb  equ     05h
        0010                        wd1797  equ     10h
        001E                        kbdat   equ     1eh
        0019                        ctc1    equ     x'19'           ;ctc1 port address

                                    ;       Other Equates
                                    ;
        0081                        encntr  equ     x'81'           ;enable ctc command
        0001                        stcntr  equ     x'01'           ;stop ctc command
        0000                        rev0    equ     x'00'           ;4.00 Revision Level
        0001                        rev1    equ     x'01'           ;4.01 Revision Level
        0064                        rev50   equ     5*100-400       ;5.00 Revision level
        003C                        cnfgoff equ     x'3c'           ;monitor configuration offset
        0006                        cnfbyte equ     x'06'           ;configuration subroutine byte offset
        0008                        kblp    equ     x'08'           ;configuration bit id for LPKYBD
        0008                        romofs  equ     x'08'           ;PROMPT offset between 4.02 & 4.01 monitor
        0001                        lpkofs  equ     x'01'           ;additonal sector required for table storage
        007B                        upper   equ     'z'+1           ;upper limit for alpha test
        0061                        lower   equ     'a'             ;lower limit for alpha test
        0020                        upascii equ     'a'-'A'         ;set to upper case ASCII mask
        0000                        zero    equ     0               ;zero
        00FF                        setflg  equ     x'ff'           ;set flag

                                    ;       Equates
                                    ;
        0004                        c.five  equ     04
        0006                        c.sasi  equ     06
        0001                        o.term  equ     0001h
        0300                        sasidl  equ     300h

                                    ;       Internal equates
                                    ;
        001D                        rtab1   equ     29              ;rigid disk tables sector 1
        001E                        rtab2   equ     30              ;  "    "     "      "    2
        0004                        ftab1   equ     04              ;floppy "     "      "    1
        0005                        ftab2   equ     05              ;  "    "     "      "    2
```

```
0006                    ftab3   equu    06              ;  "    "     "    "    3
0000                    lang    equ     00              ;offset of language no. in index table
0001                    kbrd    equ     01              ;offset of k/b flag in index table
0002                    font    equ     02              ;offset of font flag in index table
0003                    prnt    equ     03              ;offset of printer flag in index table
0004                    kbrdtb  equ     04              ;offset of k/b tables in first sector
001A                    clrs    equ     1ah             ;clear screen
001B                    esc     equ     1bh             ;escape key
0004                    eot     equ     04h             ;end of text
000D                    cr      equ     0dh             ;carriage return
000A                    lf      equ     0ah             ;line feed

                                subttl  RX1984 Restart
                                page
```

```
        0000'                       start:
                                            .phase  rx1984

                                    ;;      RX1984
                                    ;       Entry here from Xerox monitor bfore entering SIGNON.
                                    ;
                                    ;       Input:-
                                    ;               hl - cmdtab
                                    ;               de - seltab
                                    ;               bc - cloc
                                    ;
                                    ;
        1800    C5                          push    bc
        1801    D5                          push    de
        1802    E5                          push    hl
        1803    21 0000                     ld      hl,0
        1806    CD F03C                     call    config          ;get monitor configuration
        1809    7C                          ld      a,h
        180A    FE 00                       cp      rev0
        180C    CA 187E                     jp      z,noload        ;skip if below 4.01
        180F    FE 64                       cp      rev50
        1811    D2 187E                     jp      nc,noload       ;skip if 5.00 or above
        1814    21 1B1A                     ld      hl,rv1tbl       ;4.01 spring board table
        1817    FE 01                       cp      rev1
        1819    28 03                       jr      z,tbxfer        ;skip if 4.01
        181B    21 1B29                     ld      hl,rv2tbl       ;4.02+ spring board table
        181E    11 F06C             tbxfer: ld      de,mntrex
        1821    01 000F                     ld      bc,jtblsz
        1824    F5                          push    af              ;save monitor level
        1825    ED B0                       ldir                    ;append monitor table with lpkybd jmp vectors
        1827    DD 2A F03D                  ld      ix,(monitr+cnfgoff+1)    ;set address at monitor config:
        182B    DD 7E 06                    ld      a,(ix+cnfbyte)
        182E    F6 08                       or      kblp            ;set low profile bit flag
        1830    DD 77 06                    ld      (ix+cnfbyte),a
        1833    F1                          pop     af              ;recover monitor level

                                    ;;      Alter BOOT commnd vectors
                                    ;
        1834    DD E1                       pop     ix              ;cmdtab address
        1836    DD E5                       push    ix
        1838    DD 36 02 3D'                ld      (ix+boff1),low rxboot           ;assume 4.01 monitor
        183C    DD 36 03 06'                ld      (ix+boff1+1),high rxboot
        1840    DD 36 18 3D'                ld      (ix+boff2),low rxboot
        1844    DD 36 19 06'                ld      (ix+boff2+1),high rxboot
        1848    FE 01                       cp      rev1            ;monitor check
        184A    28 10                       jr      z,soout         ;skip if 4.01 monitor
        184C    DD 36 02 45'                ld      (ix+boff1),low (rxboot+romofs)  ;4.02+ monitor boot over addr
        1850    DD 36 03 06'                ld      (ix+boff1+1),high (rxboot+romofs)
        1854    DD 36 18 45'                ld      (ix+boff2),low (rxboot+romofs)
        1858    DD 36 19 06'                ld      (ix+boff2+1),high (rxboot+romofs)

                                    ;;      Alter keyboard interrupt service
                                    ;
```

```
1 85C    DD 2A FF1A          Soout:  ld      ix,(kbvec)      ;k/b int vector
1860     DD 36 0A CD                 ld      (ix+kboff),0cdh ;CALL operation
1864     DD 36 0B 6C                 ld      (ix+kboff+1),low kybdlp
1868     DD 36 0C F0                 ld      (ix+kboff+2),high kybdlp

                             ;;      Move in RX SIGNON to o/l area and execute it
                             ;
186C     E1                          pop     hl
186D     D1                          pop     de
186E     C1                          pop     bc
186F     C1                          pop     bc              ;throw away  return address
1870     21 0552'                    ld      hl,rxsign       ;rom address
1873     11 FC5D                     ld      de,tca          ;o/l area
1876     01 00EB                     ld      bc,rxsigl       ;length
1879     ED B0                       ldir
187B     C3 FC5D                     jp      tca             ;GO SIGN ON
187E     E1                  noload: pop     hl
187F     D1                          pop     de
1880     C1                          pop     bc
1881     3E FF                       ld      a,x'ff'         ;wrong monitor
1883     A7                          and     a               ;load signon from monitor
1884     C9                          ret

                                     subttl  ROM resident CRT Driver
                                     page
```

```
                                ;;      Crtdvr - Crt Driver RX Addition.
                                ;
18B5    2A FFAC                 Rxcrt:  ld      hl,(cursor)     ;set cursor address
18B8    3A FFB4                         ld      a,(chrsav)      ;retrieve character under cursor
18BB    77                              ld      (hl),a          ;replace character under cursor
18BC    32 F319                         ld      (gold),a        ;bury balcones gold
18BF    3A FFB2                         ld      a,(leadin)      ;set leadin state
1892    B7                              or      a
1893    C2 0196                         jp      nz,crtd2        ;if processing escape sequence
1896    3A F0E3                         ld      a,(mask)        ;get keyboard mask
1899    A1                              and     c
189A    4F                              ld      c,a
189B    FE 20                           cp      ' '
189D    DA 0196                         jp      c,crtd2         ;if control code
18A0    CD 18A6                         call    fonchk          ;do font translation
18A3    C3 0182                         jp      crtd1           ;go to XR code

                                ;;      Subroutine fonchk does the font translation for national
                                ;       character sets.
                                ;       entry:  C contains the character
                                ;       exit:   C contains the translation
                                ;
18A6    E5                      Fonchk: push    hl              ;save cursor posn.
18A7    79                              ld      a,c             ;get char in a
18A8    E6 80                           and     10000000b       ;preserve attribute bit
18AA    F5                              push    af
18AB    21 FFB3                         ld      hl,attrib       ;point to attribute enabled flag
18AE    B6                              or      (hl)            ;test if set
18AF    28 0A                           jr      z,fon1          ;no attribute bit - go do translation
18B1    11 01DD                         ld      de,grpad        ;check if graphics mode
18B4    2A F339                         ld      hl,(prvatt)     ;current attribute mode
18B7    ED 52                           sbc     hl,de
18B9    28 0F                           jr      z,fon2          ;grahics mode - no translate
18BB    79                      fon1:   ld      a,c             ;here to do translate
18BC    CB BF                           res     7,a             ;clear attribute bit
18BE    21 F960                         ld      hl,fontbl       ;address of exceptions table
18C1    01 000D                         ld      bc,fontsz       ;size of exceptions table
18C4    ED B1                           cpir                    ;search for char. in exceptions
18C6    4F                              ld      c,a             ;restore char to c
18C7    CC 18CF                         call    z,fntran        ;if found do translation
18CA    F1                      fon2:   pop     af              ;retrieve attribute bit
18CB    B1                              or      c               ;or it in
18CC    4F                              ld      c,a
18CD    E1                              pop     hl              ;retrieve cursor
18CE    C9                              ret

                                ;;      s/r fntran translates font characters
                                ;       entry: (HL) - address+1 of char to be translated in fontbl
                                ;       exit:  (c)  - translated character
                                ;
18CF    2B                      Fntran: dec     hl              ;back to byte to be translated
18D0    01 000D                         ld      bc,fontsz       ;size of table
18D3    09                              add     hl,bc           ;add to address of char. to be translated
```

```
18D4   4E                 ld      c,(hl)          ;get translated value
18D5   C9                 ret

                  ;;      This routine is called from the Xerox screenprint handler. It
                  ;       picks up the character to be printed from the crt ram, does a
                  ;       reverse font translate,replaces any control codes with a space,
                  ;       does a printer translate and outputs the character to the printer.
                  ;       entry:- HL - address of byte to be printed
                  ;
18D6   E5         scrprt: push    hl
18D7   C5                 push    bc
18D8   7E                 ld      a,(hl)          ;byte for printing
18D9   CB BF              res     7,a             ;ignore attribute bit
                                                  ;do reverse font translate
18DB   21 F96D            ld      hl,fontbl+fontsz        ;point to translates
18DE   01 000D            ld      bc,fontsz
18E1   ED B1              cpir                    ;search for char.
18E3   20 07              jr      nz,scr01        ;not in table
                                                  ;in table convert to media code
18E5   01 000D            ld      bc,fontsz       ;offset back to media code
18E8   37                 scf
18E9   ED 42              sbc     hl,bc           ;points to media code
18EB   7E                 ld      a,(hl)
18EC              scr01:                          ;here with media code
18EC   FE 20              cp      20h             ;is it a control code
18EE   30 02              jr      nc,scr02        ;no
18F0   3E 20              ld      a,20h           ;yes. substitute a space
18F2   CD 1966    scr02:  call    potran          ;do printer translation
18F5   20 0D              jr      nz,scr03        ;no translation done. go output char.
                                                  ;translation done.check escape bit
18F7   CB 7F              bit     7,a             ;escape bit
18F9   28 09              jr      z,scr03         ;not set go output char
18FB   4F                 ld      c,a             ;set. save char.
18FC   3E 1B              ld      a,esc           ;output an escape
18FE   CD 1959            call    posout          ;output routine
1901   79                 ld      a,c             ;restore char.
1902   CB BF              res     7,a             ;clear escape bit
1904   CD 1959    scr03:  call    posout          ;print char
1907   C1                 pop     bc
1908   E1                 pop     hl
1909   C3 F232            jp      sprnt2          ;return to Xerox code

                  ;;      Exception print driver - ROM entry point
                  ;
190C   C5         Rmposend:push   bc
190D   E5                 push    hl
190E   CD 1914            call    posend
1911   E1                 pop     hl
1912   C1                 pop     bc
1913   C9                 ret

                  ;;      Posend - deals with character translation and escape
                  ;               sequences for the diablo 630
                  ;               input--- a contains char for output to channel b
1914   4F         Posend: ld      c,a
1915   3A F9A6            ld      a,(escsq)       ;in an escape sequence?
```

```
1918    B7                    or      a
1919    20 22                 jr      nz,pos04        ;yes
191B    79                    ld      a,c
191C    FE 1B                 cp      esc             ;escape char?
191E    20 07                 jr      nz,pos01        ;no
1920    CD 1959               call    posout          ;output char
1923    32 F9A6               ld      (escsq),a       ;set escape sequence flag
1926    C9                    ret
1927                  pos01:                          ;not escape char
1927    CD 1966               call    potran          ;do translation if neccessary
192A    20 0D                 jr      nz,pos03        ;wasn't neccessary
192C    CB 7F                 bit     7,a             ;escape marker set?
192E    28 09                 jr      z,pos02         ;no
1930    4F                    ld      c,a
1931    3E 1B                 ld      a,esc
1933    CD 1959               call    posout          ;output escape char
1936    79                    ld      a,c
1937    CB BF                 res     7,a             ;clear escape marker
1939                  pos02:                          ;escape marker not set
1939                  pos03:                          ;no translation
1939    CD 1959               call posout             ;output char
193C    C9                    ret
193D                  pos04:                          ;escape sequence
193D    FE FF                 cp      0ffh            ;3rd byte?
193F    20 06                 jr      nz,pos05        ;no 2nd
1941    79                    ld      a,c
1942    CD 1959               call    posout          ;output char
1945    18 0D                 jr      pos06
1947    79            pos05:  ld      a,c             ;2nd byte of escape sequence
1948    CD 1959               call    posout          ;output byte
194B    CD 1986               call    poesc           ;search escape table for char
194E    20 04                 jr      nz,pos06        ;not present---2 byte sequence
1950    3E FF                 ld      a,0ffh          ;set sequence for 3rd byte
1952    18 01                 jr      pos07
1954    AF            pos06:  xor     a               ;end of 2 byte sequence
1955    32 F9A6       pos07:  ld      (escsq),a       ;toggle escape sequence flag
1958    C9                    ret
1959    47            posout: ld      b,a
195A    CD F03F       siox1:  call    siordy
195D    CC F066               call    z,idle
1960    28 F8                 jr      z,siox1
1962    78                    ld      a,b
1963    D3 05                 out     (siodpb),a
1965    C9                    ret

                      ;;      Potran - does printer translation if neccessary and returns
                      ;               a flag to indicate if translation has been done.
                      ;               input- a  char for translaation
                      ;
                      ;               output- a  (translated) char
                      ;                       z  set if char is translated (otherwise reset)
                      ;
1966    21 F97A       Potran: ld      hl,prntbl       ;print exceptions table
1969    01 0016               ld      bc,prntsz       ;size of table
196C    ED B1                 cpir
196E    C0                    ret     nz              ;no match - don't translate
```

```
196F    2B                          dec     hl              ;translate
1970    01 0016                     ld      bc,prntsz
1973    09                          add     hl,bc           ;offset to translation
1974    7E                          ld      a,(hl)          ;translation byte
1975    B7                          or      a               ;if zero, requires overstriking sequence
1976    20 0B                       jr      nz,ptr01        ;non-zero - go output char
                                                            ;zero - use next 2 bytes in table as sequence
1978    23                          inc     hl
1979    7E                          ld      a,(hl)          ;first byte
197A    CD 1959                     call    posout
197D    3E 08                       ld      a,08h           ;backspace
197F    CD 1959                     call    posout
1982    23                          inc     hl              ;second byte
1983    AF                  ptr01:  xor     a               ;set z for return flag
1984    7E                          ld      a,(hl)          ;get translation
1985    C9                          ret

                            ;;      Poesc - searches the escape table for a match with the char
                            ;               passed in a. if found returns with z set otherwise
                            ;               z is clear
                            ;
1986    21 198F             Poesc:  ld      hl,pesctb       ;table of escape sequences
1989    01 0007                     ld      bc,esctsz       ;size of table
198C    ED B1                       cpir
198E    C9                          ret

198F    09 0B 0C 1E         pesctb: defb    09h,0bh,0ch,1eh,1fh,16h,11h     ;630 daisy printer
1993    1F 16 11
0007                        esctsz  equ     $-pesctb


                            ;;      Function:- to deal with characters form a position
                            ;                  encoded keyboard.
                            ;       input:-    A  character read from PIO
                            ;                  CMD/STATUS byte
                            ;                       bit 7 -CMD/STATUS byte if set
                            ;                       bit 6 -upstroke flag
                            ;                       bit 5 -y axis negative (mouse)
                            ;                       bit 4 -x axis negative (mouse)
                            ;                       bit 3 -mouse active
                            ;                       bit 2 -ctrl key station active
                            ;                       bit 1 -shift key station active
                            ;                       bit 0 -lock key station active
                            ;               First data byte
                            ;                       bit 7 -Always reset
                            ;                       bits(6-0) -key station or x mouse displacement
                            ;                                  Second mouse data byte
                            ;                       bit 7 -Always reset
                            ;                       bits(6-0) -y mouse displacement
                            ;
                            ;       output:-   1) Carry set -- command byte or sequence error
                            ;                  2) Carry clear -- translated character returned in A
                            ;
1996    2F                  Pekhdl: cpl                     ;complement keyboard byte
1997    D5                          push    de              ;save registers
1998    16 00                       ld      d,zero          ;get flags
```

```
199A    CB 7F                   bit     cmd,a           ;command byte?
199C    28 17                   jr      z,kypos         ;skip to position byte handler
199E    18 01                   jr      cmdb            ;skip to command byte handler
19A0                    nochar  equ     $
19A0    7A              peknoc2:ld      a,d             ;clear command byte (non-valid position byte)
19A1    32 F9A7         cmdb:   ld      (cmdstat),a     ;save command-status byte
19A4    CB 5F                   bit     mouse,a         ;mouse cmd?
19A6    28 05                   jr      z,peknoc        ;skip if not mouse
19A8    21 F95D                 ld      hl,mstbl
19AB    CB 8E                   res     xy,(hl)
19AD    CD 1A66         peknoc: call    stpctcl         ;reset repeat flag
19B0    37              peknoc1:scf                     ;this no the position byte
19B1    D1              pekex:  pop     de              ;recover registers
19B2    C3 F9B7                 jp      lpkext          ;jmp instead of ret - small interrupt stack
19B5    5F              kypos:  ld      e,a             ;save position code
19B6    21 F9A7                 ld      hl,cmdstat      ;fetch command byte
19B9    CB 7E                   bit     cmd,(hl)        ;out of sync check
19BB    28 F0                   jr      z,peknoc        ;quit if no command byte
19BD    CB 5E                   bit     mouse,(hl)      ;test for mouse movement
19BF    C2 1A7D                 jp      nz,mice         ;skip if mouse moved
19C2    CB 76                   bit     ustrk,(hl)      ;test key posistion
19C4    20 20                   jr      nz,upstrk       ;jump if special upstroke
19C6    CD 19DB                 call    ctrtst          ;test for control codes
19C9    28 D5                   jr      z,peknoc2       ;quit if non printable control character
19CB    CD 19FF                 call    tblsel          ;select translation table
19CE    CD 1A23                 call    alphtst         ;test for alpha lock char
19D1    CD 1A3D                 call    rptst           ;test for repeat keys
19D4    21 F9A7         charout:ld      hl,cmdstat      ;fetch command byte
19D7    72                      ld      (hl),d          ;clear command byte (valid position byte)
19D8    A7                      and     a               ;clear carry
19D9    18 D6                   jr      pekex

                        ;;      Character is tested for the lock, shift, and ctrl key station.
                        ;
                        ;       input   a - key station code
                        ;
                        ;       output z - set if lock,shift, or ctrl key station
                        ;
19DB    E5              Ctrtst: push    hl              ;save registers
19DC    21 F953                 ld      hl,ctrltb       ;non printable char table
19DF    01 0006                 ld      bc,cntctr       ;byte count of table
19E2    ED B1                   cpir                    ;search table
19E4    E1                      pop     hl
19E5    C9                      ret

                        ;;      The up-stroke is tested for special up-stroke key-stations.
                        ;
                        ;       input   a - key station code
                        ;               strkup - user enable flag
                        ;
                        ;       output  a - translated up-stroke key-station code
                        ;
19E6    21 F95D         Upstrk: ld      hl,mstbl        ;user enable up-stroke flag
19E9    CB 5E                   bit     strkup,(hl)
19EB    28 B3                   jr      z,nochar        ;quit if user inhibited
19ED    21 F959                 ld      hl,ups          ;exception key-station table
```

```
19F0    01 0002                  ld      bc,upssz         ;byte count of table
19F3    ED B1                    cpir                     ;search table
19F5    20 A9                    jr      nz,nochar        ;quit if no match
19F7    2B                       dec     hl               ;get exception
19F8    01 0002                  ld      bc,upssz
19FB    09                       add     hl,bc
19FC    7E                       ld      a,(hl)
19FD    18 D5                    jr      charout          ;return translated character

                        ;;       The appropriate keyboard translation table is selected
                        ;
                        ;        input   hl - command-status address
                        ;                de - key station code
                        ;
                        ;        output  a  - translated key station code
                        ;
19FF    7E              Tblsel:  ld      a,(hl)           ;move cmd-status byte
1A00    E5                       push    hl               ;save command-status ptr
1A01    21 F867                  ld      hl,shtab         ;preset to shift table
1A04    CB 4F                    bit     shift,a          ;shift bit set?
1A06    20 17                    jr      nz,cmdbl         ;skip if set
1A08    21 F8CE                  ld      hl,cdtab         ;preset to control table
1A0B    CB 57                    bit     ctrl,a           ;control bit set?
1A0D    20 10                    jr      nz,cmdbl         ;skip if set
1A0F    21 F800                  ld      hl,tabl          ;preset to un shifted table
1A12    CB 47                    bit     lock,a           ;lock key set
1A14    28 09                    jr      z,cmdbl          ;skip if reset
1A16    3A F93B                  ld      a,(shftlck)      ;lock key set
1A19    A7                       and     a                ;test for shift lock (not alpha lock)
1A1A    28 03                    jr      z,cmdbl          ;skip if reset
1A1C    21 F867                  ld      hl,shtab         ;preset to shift table
1A1F                    cmdbl:                            ;here with translation table address in hl
1A1F    19                       add     hl,de            ;index into table
1A20    7E                       ld      a,(hl)           ;get translated char
1A21    E1                       pop     hl               ;recover command-status ptr
1A22    C9                       ret

                        ;;       If the lock key is depressed, the translated character is
                        ;        tested to see if it is an alphabet.  If it is lower case,
                        ;        then it is forced upper case.
                        ;
                        ;        input   hl - command-status address
                        ;                a  - translated character
                        ;
                        ;        output a  - translated character(upper chase if alpha+lock)
                        ;
1A23    CB 46           Alphtst:bit      lock,(hl)        ;test alpha lock flag
1A25    C8                       ret     z                ;quit if not alpha lock
1A26    FE 7B                    cp      upper            ;test for upper alpha range
1A28    30 06                    jr      nc,alphexc       ;skip if non alpha range
1A2A    FE 61                    cp      lower            ;test for lower case alpha range
1A2C    38 02                    jr      c,alphexc        ;skip if not lower alpha case
1A2E    D6 20                    sub     upascii          ;set upper case ASCII alpha character

                        ;;       Three additional caracters are allowed for the alpha lock key
                        ;
```

```
                               ;       input   hl - command-status address
                               ;               a  - translated character
                               ;
                               ;       output  a  - upper case exception
                               ;
1A30    21 F935                Alphexc:ld      hl,captab               ;lock exception table
1A33    01 0003                        ld      bc,cptbsz               ;table size
1A36    ED B1                          cpir                            ;search
1A38    C0                             ret     nz                      ;quit if not found
1A39    23                             inc     hl                      ;get exception
1A3A    23                             inc     hl
1A3B    7E                             ld      a,(hl)
1A3C    C9                             ret

                               ;;      Checks for repeat character.  If repeat character, the millisec
                               ;       timer is vector address is modified and the timer is set up
                               ;       for 0.5 second.  The timer is kicked off.
                               ;
                               ;       input   a - translated character
                               ;
1A3D    21 F940                Rptst:  ld      hl,rptbl        ;repeat char table
1A40    01 0013                        ld      bc,cntrp        ;number of repeat chars
1A43    ED B1                          cpir                    ;test for repeat chars
1A45    C0                             ret     nz              ;quit if not repeat char
1A46    2A F93C                        ld      hl,(tick)       ;millisec count
1A49    22 F9A8                        ld      (millcnt),hl    ;save it in table
1A4C    21 F9AA                        ld      hl,rptchar      ;repeat char save address
1A4F    77                             ld      (hl),a          ;save repeat char
1A50    F5                             push    af
1A51    23                             inc     hl              ;repeat flag address
1A52    36 FF                          ld      (hl),setflg     ;set repeat flag
1A54    2A FF12                        ld      hl,(ctcvec+2)   ;get 1 millisec interrupt vector
1A57    22 F9AC                        ld      (save),hl       ;save it
1A5A    21 F9D0                        ld      hl,rptclk       ;kybd repeat key timer
1A5D    22 FF12                        ld      (ctcvec+2),hl   ;substitute it
1A60    3E 81                          ld      a,encntr        ;enable millisec timmer
1A62    D3 19                          out     (ctc1),a        ;do it
1A64    F1                             pop     af              ;recover character
1A65    C9                             ret

                               ;;      This routine stops the millisecond timer and restores the
                               ;       original timer vector
                               ;
1A66    21 F9AB                Stpctc1:ld      hl,rptflg       ;fetch repeat char flag
1A69    7E                             ld      a,(hl)
1A6A    A7                             and     a               ;set flags
1A6B    C8                             ret     z               ;quit if no repeat keys
1A6C    72                             ld      (hl),d          ;clear repeat char flag
1A6D    2A F9AC                        ld      hl,(save)       ;original 1 millisec interrupt address
1A70    22 FF12                        ld      (ctcvec+2),hl   ;restore it
1A73    3A F20E                        ld      a,(spact)       ;fetch screen print flag
1A76    A7                             and     a
1A77    C0                             ret     nz              ;don't kill timer, if screen printing
1A78    3E 01                          ld      a,stcntr        ;stop timer
1A7A    D3 19                          out     (ctc1),a
1A7C    C9                             ret
```

```

                                   ;;      This is the optical mouse handler.  The delta x and y mouse
                                   ;       movement is accumulate and stored in memory locations.
                                   ;
                                   ;       input   a=      mouse delta either x or y
                                   ;               xy=     zero for x mouse delta
                                   ;                       x'ff' for y mouse delta
                                   ;               hl=     cmdstat address
                                   ;               cmdstat=bit 3   mouse moved
                                   ;                       bit 4   x delta negative
                                   ;                       bit 5   y delta negative
                                   ;               mstbl  =bit 7   mouse enabled user flag
                                   ;                       bit 6   intrp/npol
                                   ;                       bit 1   y/nx byte
                                   ;                       bit 0   mouse table is updated
                                   ;               mbyte  =interrupt return byte
                                   ;               xmax   =maximum horizontal display units
                                   ;               ymax   =maximum vertical display units
                                   ;
                                   ;       output mstbl=   bit 0, mouse table updated
                                   ;               xloc=   x position of mouse
                                   ;               yloc=   y position of mouse
                                   ;               dxmv=   prior x signed displacement
                                   ;               dymv=   prior y signed displacement
                                   ;
                                   ;       special requirements
                                   ;               The majority of the mouse-pointer table is require memory
                                   ;               resident in the in the user address space above X'BFFF'
                                   ;               since this handle is ROM resident.  Variables MSTBL AND MSPTR
                                   ;               reside in keyboard RAM space.  Variable MSPTR pointers to
                                   ;               where the user mouse table resides.  These variable are
                                   ;               update by both the handler and the user.  The following
                                   ;               data structure resides in the user's RAM space only if
                                   ;               the two-button mouse pointer is required for the applicaton:
                                   ;               (MSPTR+0)       =MBYTE
                                   ;               (MSPTR+1)       =XMAX-LSB
                                   ;               (MSPTR+2)       =    -MSB
                                   ;               (MSPTR+3)       =YMAX-LSB
                                   ;               (MSPTR+4)       =    -MSB
                                   ;               (MSPTR+5)       =XLOC-LSB
                                   ;               (MSPTR+6)       =    -MSB
                                   ;               (MSPTR+7)       =YLOC-LSB
                                   ;               (MSPTR+8)       =    -MSB
                                   ;               (MSPTR+9)       =DXMV-LSB
                                   ;               (MSPTR+A)       =    -MSB
                                   ;               (MSPTR+B)       =DYMV-LSB
                                   ;               (MSPTR+C)       =    -MSB
                                   ;
1A7D    5F                         Mice:   ld      e,a             ;save mouse delta
1A7E    7E                                 ld      a,(hl)
1A7F    47                                 ld      b,a             ;save cmd/status byte
1A80    21 F95D                            ld      hl,mstbl        ;mouse table
1A83    CB 7E                              bit     msflg,(hl)      ;test for user enabled
1A85    CA 19A0                            jp      z,nochar        ;quit if mouse handler is not enabled
1A88    DD E5                              push    ix              ;save register
1A8A    DD 2A F95E                         ld      ix,(msptr)      ;fetch user's table
```

```
1A8E   CB 4E                      bit    xy,(hl)          ;test y/nx mouse byte flag
1A90   20 26                      jr     nz,micey         ;skip if y axis delta
1A92   DD 6E 05                   ld     l,(ix+5)         ;fetch current x position
1A95   DD 66 06                   ld     h,(ix+6)
1A98   7B                         ld     a,e              ;save unsigned mouse delta byte
1A99   CB 60                      bit    xneg,b           ;test polarity of x delta
1A9B   CD 1AEB                    call   mice1            ;add delta & do min value check
1A9E   DD 73 09                   ld     (ix+x'9'),e      ;save signed mouse delta word
1AA1   DD 72 0A                   ld     (ix+x'a'),d
1AA4   DD 5E 01                   ld     e,(ix+1)         ;fetch max position value
1AA7   DD 56 02                   ld     d,(ix+2)
1AAA   CD 1B00                    call   mice2            ;do max value check
1AAD   DD 75 05                   ld     (ix+5),l         ;save position
1AB0   DD 74 06                   ld     (ix+6),h
1AB3   DD E1             micex1:  pop    ix               ;restore regiser
1AB5   C3 19B0                    jp     peknocl          ;return to wait for y mouse byte
1AB8   DD 6E 07          micey:   ld     l,(ix+7)         ;fetch current y position
1ABB   DD 66 08                   ld     h,(ix+8)
1ABE   7B                         ld     a,e              ;save unsigned mouse delta byte
1ABF   CB 68                      bit    yneg,b           ;test polarity of y delta
1AC1   CD 1AEB                    call   mice1            ;add delta & do min value check
1AC4   DD 73 0B                   ld     (ix+x'b'),e      ;save signed mouse delta word
1AC7   DD 72 0C                   ld     (ix+x'c'),d
1ACA   DD 5E 03                   ld     e,(ix+3)         ;fetch max position value
1ACD   DD 56 04                   ld     d,(ix+4)
1AD0   CD 1B00                    call   mice2            ;do max value check
1AD3   DD 75 07                   ld     (ix+7),l         ;save position
1AD6   DD 74 08                   ld     (ix+8),h
1AD9   21 F95D                    ld     hl;mstbl         ;update mouse status
1ADC   CB C6                      set    msmov,(hl)       ;set the mouse update flag
1ADE   CB 76                      bit    mintrp,(hl)      ;test for interrupt mode
1AE0   CA 1AB3                    jp     z,micex1         ;bye bye if polling mode
1AE3   DD 7E 00                   ld     a,(ix)           ;user interrupt byte
1AE6   DD E1                      pop    ix               ;restore regiser
1AE8   C3 19D4                    jp     charout          ;bye bye

                         ;;       This routine adds the delta to either the x or y position and
                         ;        does minimum position check
                         ;        input   e=      mouse delta (absolute)
                         ;                zero    set if positive mouse delta
                         ;                        reset if negative mouse delta
                         ;
                         ;        output  hl=     update position
                         ;
1AEB   20 04             Mice1:   jr     nz,mice11        ;skip if delta negative
1AED   16 00                      ld     d,zero           ;set msb positive
1AEF   18 07                      jr     mice12
1AF1   16 FF             mice11:  ld     d,-1             ;set msb negative
1AF3   7B                         ld     a,e              ;recover delta
1AF4   2F                         cpl                     ;2's complement
1AF5   C6 01                      add    a,1
1AF7   5F                         ld     e,a              ;put negative value back
1AF8   19                mice12:  add    hl,de            ;add delta to mouse position
1AF9   7C                         ld     a,h              ;get msb
1AFA   A7                         and    a
1AFB   F0                         ret    p                ;skip if msb is positive
```

```
1AFC    21 0000                   ld      hl,zero         ;set to minimum boundary
1AFF    C9                        ret

                         ;;       This routine does a maximum x or y position check
                         ;        input   hl=     x or y position
                         ;                de=     x or y max value
                         ;
                         ;        output  a =     mstbl
                         ;
1B00    7A               Mice2:   ld      a,d             ;msb position test
1B01    BC                        cp      h
1B02    38 04                     jr      c,mice21        ;skip if msb too big
1B04    7B                        ld      a,e             ;lsb position test
1B05    BD                        cp      l
1B06    30 01                     jr      nc,mice22       ;skip if lsb is not too big
1B08    EB               mice21:  ex      de,hl           ;force maximum limit
1B09    3A F95D          mice22:  ld      a,(mstbl)       ;mouse table
1B0C    CB 4F                     bit     xy,a            ;complement xy flag
1B0E    28 04                     jr      z,mice23
1B10    CB 8F                     res     xy,a
1B12    18 02                     jr      mice24
1B14    CB CF            mice23:  set     xy,a
1B16    32 F95D          mice24:  ld      (mstbl),a       ;update table
1B19    C9                        ret

                         ;;       Jump table for keyboard translator and interrupt handler.
                         ;        Exit points and monitor adjustment points for the SIGNON
                         ;        overlay and boot overlay
                         ;
1B1A    C3 F9AF          Rv1tbl:  jp      lpkybd          ;4.01   monitor lpkybd jump table
1B1D    C3 F167                   jp      mkey2
1B20    C3 F18F                   jp      mkey5
1B23    C3 FC3D                   jp      mpnext
1B26    C3 FA95                   jp      mprmt0
000F                     jtblsz   equ     $-rv1tbl

1B29    C3 F9AF          rv2tbl:  jp      lpkybd          ;4.02   monitor lpkybd jump table
1B2C    C3 F167                   jp      mkey2
1B2F    C3 F18F                   jp      mkey5
1B32    C3 FC45                   jp      mpnext+romofs
1B35    C3 FA9D                   jp      mprmt0+romofs

                         ;;       The keyboard tables are restored to the original default values
                         ;        that are stored in rom
                         ;
1B38    C5               Movtbl:  push    bc
1B39    D5                        push    de
1B3A    E5                        push    hl
1B3B    21 034A'                  ld      hl,tables
1B3E    11 F800                   ld      de,tabled
1B41    01 0159                   ld      bc,tablex
1B44    ED B0                     ldir
1B46    E1                        pop     hl
1B47    D1                        pop     de
1B48    C1                        pop     bc
1B49    C9                        ret
```

```
                                .dephase
                                subttl  RAM resident (Tables)
                                page
```

```
        034A'                              tables:                          ;rom address
                                                  .phase  tabled            ;ram address

        0010                                      .radix  16

                                           ;;     k/b unshifted table
                                           ;
        F800    00 1B 31 32                Tabl:  defb    00h,1bh,31h,32h,33h,34h,35h,36h        ;nul,esc,1,2,3,4,5,6
        F804    33 34 35 36
        F808    37 38 39 30                       defb    37h,38h,39h,30h,2dh,3dh,08h,09h        ;7,8,9,0,-,=,bs,tab
        F80C    2D 3D 08 09
        F810    71 77 65 72                       defb    71h,77h,65h,72h,74h,79h,75h,69h        ;q,w,e,r,t,y,u,i
        F814    74 79 75 69
        F818    6F 70 5B 5D                       defb    6fh,70h,5bh,5dh,0dh,0ee,61h,73h        ;o,p,[,],cr,lctrl,a,s
        F81C    0D EE 61 73
        F820    64 66 67 68                       defb    64h,66h,67h,68h,6ah,6bh,6ch,3bh        ;d,f,g,h,j,k,l,;
        F824    6A 6B 6C 3B
        F828    27 0A EC 2E                       defb    27h,0ah,0ec,2eh,7ah,78h,63h,76h        ;',lf,lshift,.,z,x,c,v
        F82C    7A 78 63 76
        F830    62 6E 6D 2C                       defb    62h,6eh,6dh,2ch,2eh,2fh,0ed,1eh        ;b,n,m,,,.,/,rshift,help
        F834    2E 2F ED 1E
        F838    EF 20 EB F1                       defb    0ef,20h,0eb,0f1,0f2,0f3,0f4,0f5        ;rctrl,sp,f1,f2,f3,f4,f5
        F83C    F2 F3 F4 F5
        F840    F6 F7 F8 F9                       defb    0f6,0f7,0f8,0f9,0fa,0fb,0fc,37h        ;f6,f7,f8,f9,f10,f11,f12,7
        F844    FA FB FC 37
        F848    38 39 2C 34                       defb    38h,39h,2ch,34h,35h,36h,0bd,31h        ;8,9,,,4,5,6,=enter,1
        F84C    35 36 BD 31
        F850    32 33 30 E7                       defb    32h,33h,30h,0e7,82h,84h,83h,80h        ;2,3,0,next,darr,larr,rarr,h
        F854    82 84 83 80
        F858    81 E6 FD 7F                       defb    81h,0e6,0fd,7fh,2bh,2dh,2ah,2fh        ;uarr,prev,acc,del,+,-,mul,d
        F85C    2B 2D 2A 2F
        F860    F0 18 8E 8F                       defb    0f0,18h,8eh,8fh,0a0,0a2,0a4            ;ins,can,msw1,msw2,rx1,rx2,r
        F864    A0 A2 A4

                                           ;;     k/b shifted
                                           ;
        F867    00 1B 21 40                Shtab: defb    00h,1bh,21h,40h,23h,24h,25h,5eh        ;nul,esc,!,@,#,$,%,^
        F86B    23 24 25 5E
        F86F    26 2A 28 29                       defb    26h,2ah,28h,29h,5fh,2bh,08h,09h        ;&,*,(,),_,+,bs,tab
        F873    5F 2B 08 09
        F877    51 57 45 52                       defb    51h,57h,45h,52h,54h,59h,55h,49h        ;Q,W,E,R,T,Y,U,I
        F87B    54 59 55 49
        F87F    4F 50 7B 7D                       defb    4fh,50h,7bh,7dh,0dh,0ee,41h,53h        ;O,P,{,},cr,lcrtl,A,S
        F883    0D EE 41 53
        F887    44 46 47 48                       defb    44h,46h,47h,48h,4ah,4bh,4ch,3ah        ;D,F,G,H,J,K,L,:
        F88B    4A 4B 4C 3A
        F88F    22 0A EC 2E                       defb    22h,0ah,0ec,2eh,5ah,58h,43h,56h        ;",lf,lshift,.,Z,X,C,V
        F893    5A 58 43 56
        F897    42 4E 4D 3C                       defb    42h,4eh,4dh,3ch,3eh,3fh,0ed,1eh        ;B,N,M,<,>,?,rshift,help
        F89B    3E 3F ED 1E
        F89F    EF 20 EB F1                       defb    0ef,20h,0eb,0f1,0f2,0f3,0f4,0f5        ;rctrl,sp,lock,f1,f2,f3,f4,f
        F8A3    F2 F3 F4 F5
        F8A7    F6 F7 F8 F9                       defb    0f6,0f7,0f8,0f9,0fa,0fb,0fc,37h        ;f6,f7,f8,f9,f10,f11,f12,7
        F8AB    FA FB FC 37
```

```
F8AF    38 39 2C 34                       defb   38h,39h,2ch,34h,35h,36h,0bd,31h      ;8,9,,,4,5,6,=enter,1
F8B3    35 36 BD 31
F8B7    32 33 30 E7                       defb   32h,33h,30h,0e7,82h,84h,83h,80h      ;2,3,0,next,darr,larr,rarr,h
F8BB    82 84 83 80
F8BF    81 E6 FD 7F                       defb   81h,0e6,0fd,7fh,2bh,2dh,2ah,2fh      ;uarr,prev,acc,del,+,-,mul,d
F8C3    2B 2D 2A 2F
F8C7    F0 18 8E 8F                       defb   0f0,18h,8eh,8fh,0a1,0a3,0a5          ;ins,can,msw1,msw2,rx1,rx2,r
F8CB    A1 A3 A5

                                 ;;       k/b coded
                                 ;
F8CE    00 9B 91 92              Cdtab:   defb   00h,9bh,91h,92h,93h,94h,95h,96h      ;nul,esc,1,2,3,4,5,6
F8D2    93 94 95 96
F8D6    97 98 99 90                       defb   97h,98h,99h,90h,1fh,9ah,88h,89h      ;7,8,9,0,-,=,bs,tab
F8DA    1F 9A 88 89
F8DE    11 17 05 12                       defb   11h,17h,05h,12h,14h,19h,15h,09h      ;q,w,e,r,t,y,u,i
F8E2    14 19 15 09
F8E6    0F 10 1B 1D                       defb   0fh,10h,1bh,1dh,8dh,0ee,01h,13h      ;o,p,[,],cr,lctrl,a,s
F8EA    8D EE 01 13
F8EE    04 06 07 08                       defb   04h,06h,07h,08h,0ah,0bh,0ch,7eh      ;d,f,g,h,j,k,l,~
F8F2    0A 0B 0C 7E
F8F6    60 8A EC AE                       defb   60h,08a,0ec,0ae,1ah,18h,03h,16h      ;`,lf,lshift,.,z,s,c,v
F8FA    1A 18 03 16
F8FE    02 0E 0D 1C                       defb   02h,0eh,0dh,1ch,7ch,5ch,0ed,9eh      ;b,n,m,,,|,\,rshift,help
F902    7C 5C ED 9E
F906    EF 00 EB D1                       defb   0ef,00h,0eb,0d1,0d2,0d3,0d4,0d5      ;rctrl,sp,lock,f1,f2,f3,f4,f
F90A    D2 D3 D4 D5
F90E    D6 D7 D8 D9                       defb   0d6,0d7,0d8,0d9,0da,0db,0dc,0b7      ;f6,f7,f8,f9,f10,f11,f12,7
F912    DA DB DC B7
F916    B8 B9 AC B4                       defb   0b8,0b9,0ac,0b4,0b5,0b6,0fe,0b1      ;8,9,,,4,5,6,=enter,1
F91A    B5 B6 FE B1
F91E    B2 B3 B0 C7                       defb   0b2,0b3,0b0,0c7,02h,04h,03h,1eh      ;2,3,0,next,darr,larr,rarr,h
F922    02 04 03 1E
F926    01 C6 DD FF                       defb   01h,0c6,0dd,0ff,0ab,0ad,0aa,0af      ;uarr,prev,acc,del,+,-,mul,d
F92A    AB AD AA AF
F92E    D0 DE 8E 8F                       defb   0d0,0de,8eh,8fh,0c8,0c9,0ca          ;ins,can,msw1,msw2,rx1,rx2,r
F932    C8 C9 CA

000A                                      .radix 10

F935    00 00 00                 captab:  defb   0,0,0            ;table of exceptions requiring shifting for
F938    00 00 00                 cptex:   defb   0,0,0            ;caps lock key.(3 excepts then 3 translates)
0003                             cptbsz   equ    ($-captab)/2     ;size of table
F93B    00                       shftlck: defb   0                ;if set, locks all keys to shift table if lock set
F93C    F4                       tick:    defb   low hlfsec       ;lsb - repeat char speed
F93D    01                                defb   high hlfsec      ;msb
F93E    3F                       tock:    defb   low tenths       ;lsb
F93F    00                                defb   high tenths      ;msb
01F4                             hlfsec   equ    500              ;0.5 second count
003F                             tenths   equ    63               ;16 chars/sec

F940    08 0A 0D 20              rptbl:   defb   x'08',x'0a',x'0d',x'20'      ;bs,lf,cr,sp
F944    2D 2E 2F                          defb   x'2d',x'2e',x'2f'            ;-,.,/
F947    3D 58 78 7F                       defb   x'3d',x'58',x'78',x'7f'      ;=,X,x,del
F94B    81 82 83 84                       defb   x'81',x'82',x'83',x'84'      ;ucur,dcur,rcur,lcur
F94F    E0 E0 E0 E0              rptex:   defb   x'e0',x'e0',x'e0',x'e0'      ;16 TBD repeat keys
```

```
        0013                    cntrp   equ     $-rptbl

        F953    1D 2A 36 38     ctrltb: db      x'1d',x'2a',x'36',x'38',x'3a'   ;lctr,lshft,rshft,rctr,lck
        F957    3A
        F958    E0              ctrlex: db      x'e0'                           ;19 TBD additional key stations
        0006                    cntctr  equ     $-ctrltb
        0159                    tablex  equ     $-tabl

        F959    E0 E0           ups:    defb    x'e0',x'e0'     ;upstroke exception key stations
        F95B    E0 E0           upsx:   defb    x'e0',x'e0'     ;upstroke char or code translations
        0002                    upssz   equ     ($-ups)/2       ;size
        F95D    00              mstbl:  defb    0               ;mouse table
        0007                    msflg   equ     7               ;mouse translator enabled if set
        0006                    mintrp  equ     6               ;user interrupt with mbyte else user polls
        0003                    strkup  equ     3               ;upstroke user enable flag
        0001                    xy      equ     1               ;set for x delta else y delta
        0000                    msmov   equ     0               ;mouse table contains new data
        F95E    0000            msptr:  defw    0               ;user address containing the following table
        0160                    ktabsz  equ     $-tabl          ;size of k/b tables

                                ;;      Font translation table.
                                ;       first the exception codes
                                ;
        F960    23 3C 3E 40     Fontbl: defb    23h,3ch,3eh,40h,5bh,5ch,5dh,5eh         ;#,<,>,@,[,\,],^
        F964    5B 5C 5D 5E
        F968    60 7B 7C 7D             defb    60h,7bh,7ch,7dh,7eh                     ;`,{,|,},~
        F96C    7E

                                ;;      Now their translations
                                ;
        F96D    23 3C 3E 40             defb    23h,3ch,3eh,40h,5bh,5ch,5dh,5eh
        F971    5B 5C 5D 5E
        F975    60 7B 7C 7D             defb    60h,7bh,7ch,7dh,7eh
        F979    7E
        000D                    fontsz  equ     ($-fontbl)/2    ;size of font tables

                                ;;      Printer translation table
                                ;       first the exception codes
                                ;
        F97A    21 23 2A 2B     Prntbl: defb    21h,23h,2ah,2bh,2ch,2eh,3ch,3eh         ;!,#,*,+,,,.,<,>
        F97E    2C 2E 3C 3E
        F982    40 5B 5C 5D             defb    40h,5bh,5ch,5dh,5eh,60h,7bh,7ch         ;@,[,\,],^,`,{,|
        F986    5E 60 7B 7C
        F98A    7D 7E FF FF             defb    7dh,7eh,x'ff',x'ff',x'ff',x'ff'         ;},~,TBD,TBD,TBD,TBD
        F98E    FF FF

                                ;;      Now the translations
                                ;
        F990    21 23 2A 2B             defb    21h,23h,2ah,2bh,2ch,2eh,3ch,3eh
        F994    2C 2E 3C 3E
        F998    40 5B 5C 5D             defb    40h,5bh,5ch,5dh,5eh,60h,7bh,7ch
        F99C    5E 60 7B 7C
        F9A0    7D 7E FF FF             defb    7dh,7eh,x'ff',x'ff',x'ff',x'ff'
        F9A4    FF FF
        0016                    prntsz  equ     ($-prntbl)/2    ;size of printer table
```

```
F9A6    00              escsq:  defb    0               ;escape sequence flag
                                                        ;0=> no sequence in progress
                                                        ;1b=> expecting 2nd byte
                                                        ;ff=> expecting 3rd byte
F9A7    00              cmdstat:defb    00              ;comand-status byte
0007                    cmd     equ     7               ;command-status flag
0006                    ustrk   equu    6               ;up stroke
0005                    yneg    equ     5               ;mouse y axis negative
0004                    xneg    equ     4               ;mouse x axis negative
0003                    mouse   equ     3               ;mouse active
0002                    ctrl    equ     2               ;control key active
0001                    shift   equ     1               ;shift key active
0000                    lock    equ     0               ;lock key active
F9A8    0000            millcnt:defw    0               ;current millisecond count
F9AA    00              rptchar:defb    0               ;repeat character
F9AB    00              rptflg: defb    0               ;repeat flag
F9AC    0000            save:   defw    0               ;save address of the interrupt vector
F9AE    00              romram: defb    0               ;rom/ram memory bank status
0007                    siderom equ     7               ;ram side
01AF                    olsiz   equ     $-tabled

                        ;;      k/b interrupt handler for the low profile position encoded k/b.
                        ;       The interrupt interrupt service routine inputs two or three
                        ;       bytes from the keyboard port.  The first byte is the cmd/status
                        ;       byte.  Appropriate information is saved in memory and the return
                        ;       from interrupt code is invoked.  The second and third byte(mouse)
                        ;       is position encoded data or mouse displacement is analyzed,
                        ;       translated, and return to the existing keyboard handler if the
                        ;       information is valid; otherwise it is truncated and a return
                        ;       from interrupt is invoked.  All registers saved except for the
                        ;       A register.
                        ;
                        ;       input-  keyboard input port (data complemented) kbdat
                        ;
                        ;       output- Carry flag set - Command byte or truncated character
                        ;               Carry flag reset - Translated character in A register
                        ;
F9AF    DB 1E           Lpkybd: in      a,(kbdat)       ;read k/b port
F9B1    CD F9BE                 call    romside         ;switch to romside
F9B4    C3 1996                 jp      pekhdl          ;decode posn. enc. k/b
                                                        ;jp instead of call - interrupt stack small
F9B7    CD F9C8         lpkext: call    ramside         ;restore original memory side
F9BA    D0                      ret     nc              ;return to xerox code with char.
                                                        ;here if command byte
F9BB    C1                      pop     bc              ;throw away return address
F9BC    18 3B                   jr      rptclk2         ;return from interrupt (via xerox)

                        ;;      Romside saves the status of the current side of memory
                        ;       and switches to romside.
                        ;
                        ;       output  romram= status of syspio
                        ;
F9BE    F5              Romside:push    af              ;save register
F9BF    DB 1C                   in      a,(syspio)      ;read ram/rom status
F9C1    32 F9AE                 ld      (romram),a      ;save it
F9C4    CB FF                   set     siderom,a       ;force rom side
```

```
       F9C6    18 04                      jr      rmside2

                                  ;;      Ramside restores the ram/rom memory back to where it was
                                  ;
                                  ;       input   romram= ;status of prior ram/rom
                                  ;
       F9C8    F5                 Ramside:push    af              ;save register
       F9C9    3A F9AE                    ld      a,(romram)      ;recover prior ram/rom status
       F9CC    D3 1C              rmside2:out     (syspio),a      ;switch it
       F9CE    F1                         pop     af
       F9CF    C9                         ret

                                  ;;      This routine is the repeat key timer interrupt handler.  If the
                                  ;       the repeat flag or the count is not zero, then the return from
                                  ;       interrupt is invoked.  If the count is zero, then the repeat
                                  ;       character is passed to the ASCII keyboard handler.
                                  ;
                                  ;       input   rptflg  -repeat flag
                                  ;               rptchar -repeat char
                                  ;               millcnt -timer table
                                  ;
                                  ;       output  a       -repeat char if count is zero
                                  ;
       F9D0    ED 73 F1EC         Rptclk: ld      (savstk),sp     ;save current stack ptr
       F9D4    31 FF50                    ld      sp,intstk       ;temporary interrupt stack
       F9D7    E5                         push    hl              ;save registers
       F9D8    F5                         push    af
       F9D9    C5                         push    bc
       F9DA    3A F9AB                    ld      a,(rptflg)      ;repeat key flag
       F9DD    A7                         and     a               ;set flags
       F9DE    28 19                      jr      z,rptclk2       ;quit if not in repeat key mode
       F9E0    2A F9A8                    ld      hl,(millcnt)    ;current millisec count
       F9E3    7C                         ld      a,h
       F9E4    B5                         or      l               ;test count
       F9E5    28 06                      jr      z,rptclk1       ;skip if time elasped
       F9E7    2B                         dec     hl              ;dcr count
       F9E8    22 F9A8                    ld      (millcnt),hl    ;save  millisec count
       F9EB    18 0C                      jr      rptclk2         ;quit if not time
       F9ED    2A F93E            rptclk1:ld      hl,(tock)       ;reset millisec count
       F9F0    22 F9A8                    ld      (millcnt),hl
       F9F3    3A F9AA                    ld      a,(rptchar)     ;fetch repeat char
       F9F6    C3 F06F                    jp      key2            ;give char to keyboard key
       F9F9    C3 F072            rptclk2:jp      key5

                                  ;;      Siout - output to channel b after translation and
                                  ;               escape sequence handling
       F9FC    F3                 Rxsioo: di                      ;char in a
       F9FD    CD F9BE                    call    romside         ;switch to romside
       FA00    CD 190C                    call    rmposend        ;does real work
       FA03    CD F9C8                    call    ramside         ;restore original memory side
       FA06    FB                         ei
       FA07    C9                         ret
       FA07                       kbramend equ    $-1             ;last location of code in ram
       0208                       olsiz3  equ     $-tabled        ;size of relocatable code

                                          .dephase
```

```
 1079                                          subttl  Overlay (signon)
 1080                                          page
```

```
                                 ;;      signon - Announce System Ready.
                                 ;
0552'                            rxsign:                          ;source address in rom
                                         .phase  tca              ;execution in transient command area
FC5D    21 F091                  Signon: ld      hl,confg         ;point to configuration byte
FC60    DB 1C                            in      a,(syspio)       ;check configuration
FC62    CB 47                            bit     0,a
FC64    28 26                            jr      z,sign3          ;if SASI interface present
FC66    F3                               di
FC67    3E CF                            ld      a,11001111b      ;set Pio B in Bit Mode
FC69    D3 1D                            out     (sysctl),a
FC6B    3E 38                            ld      a,00111000b      ;turn around d0,1,2
FC6D    D3 1D                            out     (sysctl),a
FC6F    3E 80                            ld      a,10000000b      ;ensure rom switched on
FC71    D3 1C                            out     (syspio),a       ;drop all drive selects
FC73    3E D0                            ld      a,0d0h           ;reset wd-1797-02
FC75    D3 10                            out     (wd1797),a
FC77    10 FE                    sign1:  djnz    sign1            ;wait 1797 not busy
FC79    DB 1C                            in      a,(syspio)
FC7B    CB 67                            bit     c.five,a
FC7D    3E 02                            ld      a,2              ;preset 10 msec step rate
FC7F    20 04                            jr      nz,sign2         ;if not 5"
FC81    CB E6                            set     c.five,(hl)
FC83    3E 03                            ld      a,3              ;set long step
FC85    D3 10                    sign2:  out     (wd1797),a       ;restore / unload heads
FC87    32 FF54                          ld      (steprt),a
FC8A    18 1E                            jr      sign4
FC8C    CB F6                    sign3:  set     c.sasi,(hl)      ;set Sasi card installed
FC8E    21 F708                          ld      hl,Rigdpb        ;set address of rigid dpb
FC91    11 F470                          ld      de,Fivdpb        ;set address of 5.25" floppy dpb
FC94    01 0300                          ld      bc,Sasidl        ;set sasi driver length
FC97    ED B0                            ldir                     ;Move driver down
FC99    E6 02                            and     2
FC9B    20 0D                            jr      nz,sign4         ;if not A/E swap
FC9D    21 F361                          ld      hl,Seltab+1
FCA0    06 08                            ld      b,8
FCA2    7E                       sign3a: ld      a,(hl)
FCA3    EE 04                            xor     4
FCA5    77                               ld      (hl),a
FCA6    23                               inc     hl
FCA7    23                               inc     hl
FCA8    10 F8                            djnz    sign3a
FCAA    21 034A'                 sign4:  ld      hl,tables        ;move rx resident code to ram
FCAD    11 F800                          ld      de,tabled
FCB0    01 0208                          ld      bc,olsiz3
FCB3    ED B0                            ldir                     ;move on top of GETHLP
FCB5    21 FA08                          ld      hl,kbramend+1    ;next available ram loc
FCB8    22 FF3C                          ld      (availb),hl      ;tell the world
FCBB    21 0000                          ld      hl,0
FCBE    CD F03C                          call    config           ;get monitor configuration
FCC1    7C                               ld      a,h              ;monitor level
FCC2    21 10D0                          ld      hl,xrsign+sigoff-romofs ;assume 4.01 level location
FCC5    FE 01                            cp      rev1
```

```
FCC7    28 03                        jr      z,sign7         ;skip if 4.01
FCC9    21 10D8                      ld      hl,xrsign+sigoff ;4.02+ level location
FCCC    11 FCE6              sign7:  ld      de,sign6        ;put it in our signon message
FCCF    01 0004                      ld      bc,4
FCD2    ED B0                        ldir
FCD4    CD F293                      call    crtoff          ;disable rom bank
FCD7    CD F075                      call    pnext
FCDA    1A                           defb    clrs            ;clear screen
FCDB    1B 38                        defb    esc,'8'         ;set low light as default mode
FCDD    38 32 30 2D                  defm    '820-II v '
FCE1    49 49 20 76
FCE5    20
FCE6    00 00 00 00          sign6:  defb    0,0,0,0         ;*********** space for the XR rev value
FCEA    20 1F 1C 20                  defm    ' ',31,28,' 1983 Xerox Corp'
FCEE    31 39 38 33
FCF2    20 58 65 72
FCF6    6F 78 20 43
FCFA    6F 72 70
FCFD    20 28 76                     defm    ' (v'
FD00    30 31 33                     defm    ver/100+'0',(ver mod 100)/10+'0',(ver mod 10)+'0'
FD03    29 0D 0A                     defb    ')',cr,lf
FD06    0A                           defb    lf
FD07    4C 20 2D 20                  defm    'L - Load System'
FD0B    4C 6F 61 64
FD0F    20 53 79 73
FD13    74 65 6D
FD16    0D 0A                        defb    cr,lf

                                     if      o.term
FD18    48 20 2D 20                  defm    'H - Host Terminal'
FD1C    48 6F 73 74
FD20    20 54 65 72
FD24    6D 69 6E 61
FD28    6C
FD29    0D 0A                        defb    cr,lf
                                     endif

                                     if      o.term
FD2B    54 20 2D 20                  defb    'T - Typewriter'
FD2F    54 79 70 65
FD33    77 72 69 74
FD37    65 72
FD39    0D 0A                        defb    cr,lf
                                     endif

FD3B    07 04                        defb    7,eot

FD3D    CD F006              devour: call    const
FD40    CA F003                      jp      z,warm          ;go enter monitor
FD43    CD F009                      call    conin
FD46    18 F5                        jr      devour
00EB                         rxsigl  equ     $-signon

                                     .dephase
                                     subttl  Overlay (boot)
                                     page
```

```
        063D'                   rxboot:                         ;rom source address
                                        .phase  tca-romofs      ;execution address o/l area
        FC55                            ds      romofs,0        ;4.01 overlay start address
        FC5D    21 FF5D                 ld      hl,linbuf+1     ;4.02 overlay start address
        FC60    7E              boot1:  ld      a,(hl)          ;scan command line
        FC61    2C                      inc     l
        FC62    D6 0D                   sub     cr
        FC64    28 0B                   jr      z,boot2         ;if no parameter, boot from A:
        FC66    FE 13                   cp      ' '-cr
        FC68    28 F6                   jr      z,boot1         ;skip leading blanks
        FC6A    D6 34                   sub     'A'-cr
        FC6C    D8                      ret     c               ;if invalid drive
        FC6D    FE 10                   cp      16
        FC6F    3F                      ccf
        FC70    D8                      ret     c               ;if bad drive
        FC71    4F              boot2:  ld      c,a             ;set boot drive selected
        FC72    C6 41                   add     a,'A'
        FC74    32 FD72                 ld      (bootd),a       ;set up error message
        FC77    2E 00                   ld      l,0             ;set A:
        FC79    C5                      push    bc
        FC7A    E5                      push    hl
        FC7B    CD FD89                 call    swap            ;switch boot drive with A:
        FC7E    21 FD6E                 ld      hl,booter       ;set boot error return
        FC81    E5                      push    hl
        FC82    0E 00                   ld      c,0             ;then boot from A:
        FC84    CD F01B                 call    select
        FC87    C0                      ret     nz              ;if drive not configured or density error
        FC88    3E FF                   ld      a,-1
        FC8A    12                      ld      (de),a
        FC8B    11 000A                 ld      de,10           ;set dpb address offset within dph
        FC8E    19                      add     hl,de
        FC8F    5E                      ld      e,(hl)          ;set dpb address
        FC90    23                      inc     hl
        FC91    56                      ld      d,(hl)
        FC92    CD F01E                 call    home
        FC95    1A                      ld      a,(de)          ;get low sectors per track
        FC96    32 FD6D                 ld      (boots),a       ;inform boot loader
        FC99    B7                      or      a
        FC9A    20 20                   jr      nz,boot3        ;if not rigid
        FC9C    21 000D                 ld      hl,13           ;set reserved track offset within dpb
        FC9F    19                      add     hl,de
        FCA0    4E                      ld      c,(hl)          ;get reserved tracks
        FCA1    23                      inc     hl
        FCA2    46                      ld      b,(hl)
        FCA3    0B                      dec     bc              ;point behind directory
        FCA4    ED 43 FA11              ld      (phytrk),bc     ;do implied seek
                                                                ;here for rigid
        FCA8    0E 1D                   ld      c,rtabl         ;first rigid sector
        FCAA    21 ED80                 ld      hl,bootbf       ;buffer
        FCAD    CD F024                 call    read            ;layout and k/b tables
        FCB0    C0                      ret     nz
        FCB1    0E 1E                   ld      c,rtabl+lpkofs  ;2nd rigid sector
        FCB3    21 EE80                 ld      hl,bootbf+x'100' ;buffer
```

```
FCB6    CD F024                 call    read            ;layout and k/b tables
FCB9    C0                      ret     nz
FCBA    18 20                   jr      rxb01
FCBC                    boot3:                          ;here for floppy
FCBC    FE 1B                   cp      27              ;double density?
FCBE    DA FD52                 jp      c,boot4         ;no - exit
FCC1    0E 04                   ld      c,ftab1         ;first floppy sector
FCC3    21 ED80                 ld      hl,bootbf       ;buffer
FCC6    CD F024                 call    read            ;layout table and half of k/b
FCC9    C0                      ret     nz
FCCA    0E 05                   ld      c,ftab2         ;second floppy sector
FCCC    21 EE00                 ld      hl,bootbf+128
FCCF    CD F024                 call    read            ;midle third of k/b tables
FCD2    C0                      ret     nz
FCD3    0E 06                   ld      c,ftab2+lpkofs  ;third floppy sector
FCD5    21 EE80                 ld      hl,bootbf+128+128
FCD8    CD F024                 call    read            ;last third of k/b tables
FCDB    C0                      ret     nz
FCDC                    rxb01:                          ;check tables are present
FCDC    3A ED80                 ld      a,(bootbf+lang) ;language no. set?
FCDF    FE E5                   cp      0e5h
FCE1    28 6F                   jr      z,boot4         ;no - exit
FCE3    3A ED81                 ld      a,(bootbf+kbrd) ;k/b tables present?
FCE6    FE 6B                   cp      'k'
FCE8    20 68                   jr      nz,boot4        ;no - exit
FCEA    3A ED82                 ld      a,(bootbf+font) ;font tables present?
FCED    FE 66                   cp      'f'
FCEF    20 61                   jr      nz,boot4        ;no - exit
FCF1    3A ED83                 ld      a,(bootbf+prnt) ;printer tables prsent?
FCF4    FE 70                   cp      'p'
FCF6    20 5A                   jr      nz,boot4        ;no - exit
FCF8    21 ED84                 ld      hl,bootbf+kbrdtb ;move in k/b tables
FCFB    11 F800                 ld      de,tabled
FCFE    01 0160                 ld      bc,ktabsz
FD01    ED B0                   ldir
FD03    0E 1F                   ld      c,rtab2+lpkofs  ;3rd rigid sector
FD05    3A FD6D                 ld      a,(boots)       ;rigid or floppy?
FD08    B7                      or      a
FD09    28 02                   jr      z,boot5         ;rigid
FD0B    0E 07                   ld      c,ftab3+lpkofs  ;floppy - 4th sector
FD0D    21 ED80         boot5:  ld      hl,bootbf       ;buffer
FD10    CD F024                 call    read            ;font and print tables
FD13    C0                      ret     nz
FD14    3A ED80                 ld      a,(bootbf)      ;configured?
FD17    FE E5                   cp      0e5h
FD19    28 37                   jr      z,boot4         ;no -exit
FD1B    21 ED80                 ld      hl,bootbf       ;move font & print tables in
FD1E    11 F960                 ld      de,fontbl
FD21    01 0046                 ld      bc,fontsz*2+prntsz*2
FD24    ED B0                   ldir

                        ;;      alter SIOOUT
                        ;
FD26    DD 2A F019              ld      ix,(monitr+sioff);sioout address
FD2A    DD 36 00 C3             ld      (ix),0c3h       ;jump instruction
FD2E    DD 36 01 FC             ld      (ix+1),low rxsioo
```

```
FD32   DD 36 02 F9                ld      (ix+2),high rxsioo

                         ;        alter crt driver
                         ;
FD36   DD 2A F010                 ld      ix,(monitr+fcrtof)      ;address of fast crt handler
FD3A   DD 36 12 85                ld      (ix+crtcall),low rxcrt
FD3E   DD 36 13 18                ld      (ix+crtcall+1),high rxcrt
FD42   DD 21 F22F                 ld      ix,sprnt1       ;address of screenprint patch
FD46   DD 36 00 C3                ld      (ix),0c3h       ;jump instruction
FD4A   DD 36 01 D6                ld      (ix+1),low scrprt
FD4E   DD 36 02 18                ld      (ix+2),high scrprt
FD52                     boot4:                           ;here to exit
FD52   C1                         pop     bc              ;throw away booter return
FD53   C1                         pop     bc              ;get disk swap parameters
FD54   E1                         pop     hl
FD55   CD FD89                    call    swap            ;swap them back for xerox boot
FD58   C1                         pop     bc              ;throw away return address
FD59   21 0000                    ld      hl,0
FD5C   CD F03C                    call    config          ;get monitor configuration
FD5F   7C                         ld      a,h
FD60   21 11B8                    ld      hl,xrboot-romofs ;assumed 4.01 monitor boot overlay address
FD63   FE 01                      cp      rev1
FD65   28 03                      jr      z,boot6         ;skip if 4.01
FD67   21 11C0                    ld      hl,xrboot       ;address of 4.02+ monitor boot overlay
FD6A   C3 F078           boot6:   jp      prboff          ;enter xerox code to execute boot
FD6D   00                boots:   defb    0               ;workbyte to save disk type

                         ;;       Booter - Boot Error Processor.
                         ;
FD6E   CD F075           Booter:  call    pnext
FD71   07                         defb    7
FD72   64 3A 54 61       bootd:   defm    'd:Tables Load error.'
FD76   62 6C 65 73
FD7A   20 4C 6F 61
FD7E   64 20 65 72
FD82   72 6F 72 2E
FD86   04                         defb    eot
FD87   C1                         pop     bc              ;switch drives back
FD88   E1                         pop     hl

                         ;;       Swap - swap logical drives.
                         ;
                         ;        Entry:  C = first  drive index, 0-15
                         ;                L = second drive index, 0-15
                         ;
FD89   06 00             Swap:    ld      b,0             ;clear upper indices
FD8B   60                         ld      h,b
FD8C   11 F360                    ld      de,seltab       ;set select table address
FD8F   29                         add     hl,hl
FD90   19                         add     hl,de
FD91   EB                         ex      de,hl           ;set second address to DE, get seltab to HL
FD92   09                         add     hl,bc
FD93   09                         add     hl,bc           ;set first address to HL
FD94   06 02                      ld      b,2
FD96   4E                swap1:   ld      c,(hl)          ;swap two bytes
FD97   1A                         ld      a,(de)
```

```
FD98    77                         ld      (hl),a
FD99    79                         ld      a,c
FD9A    12                         ld      (de),a
FD9B    23                         inc     hl
FD9C    13                         inc     de
FD9D    10 F7                      djnz    swapl           ;if swap not complete
FD9F    C9                         ret
                                   .dephase

0788'                      romtop:
0788'                              defs    (romsiz-x'24')-(romtop-start),-1

                           ;;      Drctry is a table containing the RAM addresses of the keyboard
                           ;       tables.  This table is located on ROM side of memory.  The
                           ;       ROM address must be offset by x'1800' since resides in
                           ;       the fourth 2kx8 ROM slot.  This directory is helpful if future
                           ;       release require the RAM tables to reside in a different RAM
                           ;       location
                           ;
07DC'   F97A               Drctry: defw    prntbl          ;print exception table
07DE'   F960                       defw    fontbl          ;font exception table
07E0'   F95D                       defw    mstbl           ;mouse table
07E2'   F959                       defw    ups             ;upstroke table
07E4'   F958                       defw    ctrlex          ;function key inhibit expansion table
07E6'   F953                       defw    ctrltb          ;function key inhibit table
07E8'   F94F                       defw    rptex           ;repeat key expansion table
07EA'   F940                       defw    rptbl           ;repeat key table
07EC'   F93C                       defw    tick            ;repeat speed table
07EE'   F93B                       defw    shftlck         ;shift lock flag
07F0'   F938                       defw    cptex           ;alpha lock expansion table
07F2'   F935                       defw    captab          ;alpha lock table
07F4'   F8CE                       defw    cdtab           ;code + table
07F6'   F867                       defw    shtab           ;shift table
07F8'   F800                       defw    tabl            ;unshifted table

07FA'   0D                         defb    ver             ;revision level
07FB'   00 FF              lpid:   defb    x'00',x'ff'     ;low profile kybd id

                           ;;      define checkword to let xerox know
                           ;       that we are present
                           ;
07FD'   AA 55                      defb    0aah,55h        ;id
07FF'   00                         defb    0               ;space for checksum

                                   Subttl  Symbol Table
                                   end
```

Position encoded keyboard handler MACRO-80 3.44 09-Dec-81
Symbol Table

Macros:

Symbols:

1A30	ALPHEXC	1A23	ALPHTST	FFB3	ATTRIB
FF3C	AVAILB	0002	BOFF1	0018	BOFF2
FC60	BOOT1	FC71	BOOT2	FCBC	BOOT3
FD52	BOOT4	FD0D	BOOT5	FD6A	BOOT6
ED80	BOOTBF	FD72	BOOTD	FD6E	BOOTER
FD6D	BOOTS	0004	C.FIVE	0006	C.SASI
F935	CAPTAB	F8CE	CDTAB	19D4	CHAROUT
FFB4	CHRSAV	001A	CLRS	0007	CMD
19A1	CMDB	1A1F	CMDB1	F9A7	CMDSTAT
0006	CNFBYTE	003C	CNFGOFF	0006	CNTCTR
0013	CNTRP	F091	CONFG	F03C	CONFIG
F009	CONIN	F006	CONST	0003	CPTBSZ
F938	CPTEX	000D	CR	0012	CRTCALL
0182	CRTD1	0196	CRTD2	F293	CRTOFF
0019	CTC1	FF10	CTCVEC	0002	CTRL
F958	CTRLEX	F953	CTRLTB	19DB	CTRTST
FFAC	CURSOR	FD3D	DEVOUR	07DC	DRCTRY
0081	ENCNTR	0004	EOT	001B	ESC
F9A6	ESCSQ	0007	ESCTSZ	0010	FCRTOF
F470	FIVDPB	18CF	FNTRAN	18BB	FON1
18CA	FON2	18A6	FONCHK	0002	FONT
F960	FONTBL	000D	FONTSZ	0004	FTAB1
0005	FTAB2	0006	FTAB3	F319	GOLD
01DD	GRPAD	01F4	HLFSEC	F01E	HOME
F066	IDLE	FF50	INTSTK	000F	JTBLSZ
001E	KBDAT	0008	KBLP	000A	KBOFF
FA07	KBRAMEND	0001	KBRD	0004	KBRDTB
FF1A	KBVEC	F06F	KEY2	F072	KEY5
0160	KTABSZ	F06C	KYBDLP	19B5	KYPOS
0000	LANG	FFB2	LEADIN	000A	LF
FF5C	LINBUF	0000	LOCK	0061	LOWER
07FB	LPID	F9B7	LPKEXT	0001	LPKOFS
F9AF	LPKYBD	F0E3	MASK	1A7D	MICE
1AEB	MICE1	1AF1	MICE11	1AF8	MICE12
1B00	MICE2	1B08	MICE21	1B09	MICE22
1B14	MICE23	1B16	MICE24	1AB3	MICEX1
1AB8	MICEY	F9A8	MILLCNT	0006	MINTRP
F167	MKEY2	F18F	MKEY5	F06C	MNTREX
F000	MONITR	0003	MOUSE	1B38	MOVTBL
FC3D	MPNEXT	FA95	MPRMTO	0007	MSFLG
0000	MSMOV	F95E	MSPTR	F95D	MSTBL
19A0	NOCHAR	187E	NOLOAD	0001	O.TERM
01AF	OLSIZ	0208	OLSIZ3	19B1	PEKEX
1996	PEKHDL	19AD	PEKNOC	19B0	PEKNOC1
19A0	PEKNOC2	198F	PESCTB	FA11	PHYTRK
F075	PNEXT	1986	POESC	1927	POS01
1939	POS02	1939	POS03	193D	POS04
1947	POS05	1954	POS06	1955	POS07
1914	POSEND	1959	POSOUT	1966	POTRAN
F078	PRBOFF	0003	PRNT	F97A	PRNTBL
0016	PRNTSZ	FA62	PROMPT	F339	PRVATT
1983	PTR01	F9C8	RAMSIDE	F024	READ

```
Position encoded keyboard handler       MACRO-80 3.44   09-Dec-81
Symbol Table

0000    REV0            0001    REV1            0064    REV50
F708    RIGDPB          190C    RMPOSEND        F9CC    RMSIDE2
0008    ROMOFS          F9AE    ROMRAM          F9BE    ROMSIDE
0800    ROMSIZ          0788'   ROMTOP          F940    RPTBL
F9AA    RPTCHAR         F9D0    RPTCLK          F9ED    RPTCLK1
F9F9    RPTCLK2         F94F    RPTEX           F9AB    RPTFLG
1A3D    RPTST           001D    RTAB1           001E    RTAB2
1B1A    RV1TBL          1B29    RV2TBL          1800    RX1984
FCDC    RXB01           063D'   RXBOOT          1885    RXCRT
00EB    RXSIGL          0552'   RXSIGN          F9FC    RXSIO0
0300    SASIDL          F9AC    SAVE            F1EC    SAVSTK
18EC    SCR01           18F2    SCR02           1904    SCR03
18D6    SCRPRT          F01B    SELECT          F360    SELTAB
00FF    SETFLG          F93B    SHFTLCK         0001    SHIFT
F867    SHTAB           0007    SIDEROM         FC77    SIGN1
FC85    SIGN2           FC8C    SIGN3           FCA2    SIGN3A
FCAA    SIGN4           FCE6    SIGN6           FCCC    SIGN7
FC5D    SIGNON          0060    SIGOFF          0005    SIODPB
0019    SIOFF           F03F    SIORDY          195A    SIOX1
185C    SOOUT           F20E    SPACT           F22F    SPRNT1
F232    SPRNT2          0000'   START           0001    STCNTR
FF54    STEPRT          1A66    STPCTC1         0003    STRKUP
FD89    SWAP            FD96    SWAP1           001D    SYSCTL
001C    SYSPIO          FF18    SYSVEC          F800    TABL
F800    TABLED          034A'   TABLES          0159    TABLEX
19FF    TBLSEL          181E    TBXFER          FC5D    TCA
003F    TENTHS          F93C    TICK            F93E    TOCK
0020    UPASCII         007B    UPPER           F959    UPS
0002    UPSSZ           19E6    UPSTRK          F95B    UPSX
0006    USTRK           000D    VER             F003    WARM
0010    WD1797          0004    XNEG            11C0    XRBOOT
1078    XRSIGN          0001    XY              0005    YNEG
0000    ZERO


No Fatal error(s)
```

ALPHEXC	565	567	577#									
ALPHTST	485	562#										
ATTRIB	100#	265										
AVAILB	94#	1130										
BOFF1	45#	199	200	205	206							
BOFF2	46#	201	202	207	208							
BOOT1	1198#	1203										
BOOT2	1201	1209#										
BOOT3	1232	1250#										
BOOT4	1252	1268	1271	1274	1277	1292	1314#					
BOOT5	1285	1287#										
BOOT6	1325	1327#										
BOOTBF	84#	1242	1246	1254	1258	1262	1266	1269	1272	1275	1278	1287
	1290	1293										
BOOTD	1211	1334#										
BOOTER	1216	1332#										
BOOTS	1230	1283	1328#									
C.FIVE	132#	1102	1105									
C.SASI	133#	1110										
CAPTAB	577	895#	897	1389								
CDTAB	537	866#	1390									
CHAROUT	487#	523	718									
CHRSAV	101#	242										
CLRS	149#	1143										
CMD	460	476	972#									
CMDB	462	465#										
CMDB1	536	539	542	545	547#							
CMDSTAT	465	475	487	971#								
CNFBYTE	120#	190	192									
CNFGOFF	119#	189										
CNTCTR	500	916#										
CNTRP	593	911#										
CONFG	86#	1087										
CONFIG	57#	175	1132	1321								
CONIN	52#	1185										
CONST	51#	1183										
CPTBSZ	578	897#										
CPTEX	896#	1388										
CR	152#	1156	1162	1170	1178	1200	1202	1204				
CRTCALL	50#	1308	1309									
CRTD1	66#	254										
CRTD2	67#	247	252									
CRTOFF	76#	1141										
CTC1	110#	608	626									
CTCVEC	41#	603	606	621								
CTRL	538	977#										
CTRLEX	915#	1382										
CTRLTB	499	913#	916	1383								

CTRTST	482	498#					
CURSOR	98#	241					
DEVOUR	1183#	1186					
DRCTRY	1378#						
ENCNTR	114#	607					
EOT	151#	1181	1339				
ESC	150#	325	351	362	1144		
ESCSQ	347	354	383	967#			
ESCTSZ	426	432#					
FCRTOF	49#	1307					
FIVDPB	90#	1112					
FNTRAN	278	289#					
FON1	267	272#					
FON2	271	279#					
FONCHK	253	261#					
FONT	146#	1272					
FONTBL	274	306	934#	945	1294	1379	
FONTSZ	275	290	306	307	311	945#	1295
FTAB1	141#	1253					
FTAB2	142#	1257	1261				
FTAB3	143#	1286					
GOLD	88#	244					
GRPAD	68#	268					
HLFSEC	899	900	903#				
HOME	55#	1228					
IDLE	59#	387					
INTSTK	95#	1047					
JTBLSZ	186	774#					
KBDAT	109#	1004					
KBLP	121#	191					
KBOFF	48#	213	214	215			
KBRAMEND		1075#	1129				
KBRD	145#	1269					
KBRDTB	148#	1278					
KBVEC	43#	212					
KEY2	62#	1064					
KEY5	63#	1065					
KTABSZ	929#	1280					
KYBDLP	61#	214	215				
KYPOS	461	474#					
LANG	144#	1266					
LEADIN	99#	245					
LF	153#	1156	1157	1162	1170	1178	
LINBUF	97#	1197					
LOCK	541	562	979#				
LOWER	125#	566					
LPID	1395#						
LPKEXT	473	1008#					
LPKOFS	123#	1245	1261	1282	1286		

LPKYBD	769	776	1004#									
MASK	85#	248										
MICE	479	676#										
MICE1	690	704	728#									
MICE11	728	731#										
MICE12	730	736#										
MICE2	695	709	749#									
MICE21	751	755#										
MICE22	754	756#										
MICE23	758	761#										
MICE24	760	762#										
MICEX1	698#	715										
MICEY	685	700#										
MILLCNT	597	980#	1054	1059	1062							
MINTRP	714	924#										
MKEY2	72#	770	777									
MKEY5	73#	771	778									
MNTREX	60#	185										
MONITR	39#	51	52	53	54	55	56	57	58	59	60	61
	62	63	64	65	189	1300	1307					
MOUSE	466	478	976#									
MOVTBL	785#											
MPNEXT	80#	772	779									
MPRMTO	79#	773	780									
MSFLG	680	923#										
MSMOV	713	927#										
MSPTR	683	928#										
MSTBL	468	512	679	712	756	762	922#	1380				
NOCHAR	463#	514	518	681								
NOLOAD	178	180	228#									
O.TERM	134#	1164	1173									
OLSIZ	986#											
OLSIZ3	1076#	1127										
PEKEX	472#	490										
PEKHDL	457#	1006										
PEKNOC	467	470#	477									
PEKNOC1	471#	699										
PEKNOC2	464#	483										
PESCTB	425	430#	432									
PHYTRK	93#	1239										
PNEXT	64#	1142	1332									
POESC	378	425#										
POS01	352	356#										
POS02	360	366#										
POS03	358	367#										
POS04	349	370#										
POS05	372	376#										
POS06	375	379	382#									
POS07	381	383#										

POSEND	338	346#									
POSOUT	326	329	353	363	368	374	377	385#	413	415	
POTRAN	319	357	400#								
PRBOFF	65#	1327									
PRNT	147#	1275									
PRNTBL	400	950#	965	1378							
PRNTSZ	401	405	965#	1295							
PROMPT	78#										
PRVATT	77#	269									
PTR01	409	417#									
RAMSIDE	1008	1029#	1072								
READ	56#	1243	1247	1255	1259	1263	1288				
REV0	116#	177									
REV1	117#	182	203	1135	1324						
REV50	118#	179									
RIGDPB	91#	1111									
RMPOSEND		336#	1071								
RMSIDE2	1023	1031#									
ROMOFS	122#	205	206	207	208	779	780	1134	1195	1196	1323
ROMRAM	984#	1021	1030								
ROMSIDE	1005	1019#	1070								
ROMSIZ	38#	1369									
ROMTOP	1368#	1369									
RPTBL	592	906#	911	1385							
RPTCHAR	598	981#	1063								
RPTCLK	605	1046#									
RPTCLK1	1057	1061#									
RPTCLK2	1012	1053	1060	1065#							
RPTEX	910#	1384									
RPTFLG	615	982#	1051								
RPTST	486	592#									
RTAB1	139#	1241	1245								
RTAB2	140#	1282									
RV1TBL	181	769#	774								
RV2TBL	184	776#									
RX1984	37#	160									
RXB01	1249	1265#									
RXBOOT	199	200	201	202	205	206	207	208	1194#		
RXCRT	241#	1308	1309								
RXSIGL	225	1187#									
RXSIGN	223	1085#									
RXSIO0	1069#	1302	1303								
SASIDL	135#	1113									
SAVE	604	620	983#								
SAVSTK	40#	1046									
SCR01	309	315#									
SCR02	317	319#									
SCR03	320	323	329#								
SCRPRT	301#	1312	1313								

SELECT	54#	1219					
SELTAB	89#	1117	1350				
SETFLG	128#	602					
SHFTLCK	543	898#	1387				
SHIFT	535	978#					
SHTAB	534	546	837#	1391			
SIDEROM	985#	1022					
SIGN1	1100#	1100					
SIGN2	1104	1107#					
SIGN3	1090	1110#					
SIGN3A	1119#	1124					
SIGN4	1109	1116	1125#				
SIGN6	1138	1148#					
SIGN7	1136	1138#					
SIGNON	1087#	1187					
SIGOFF	70#	1134	1137				
SIODPB	107#	390					
SIOFF	47#	1300					
SIORDY	58#	386					
SIOX1	386#	388					
SOOUT	204	212#					
SPACT	87#	622					
SPRNT1	74#	1310					
SPRNT2	75#	332					
START	159#	1369					
STCNTR	115#	625					
STEPRT	96#	1108					
STPCTC1	470	615#					
STRKUP	513	925#					
SWAP	1215	1318	1348#				
SWAP1	1357#	1364					
SYSCTL	105#	1093	1095				
SYSPIO	106#	1020	1031	1088	1097	1101	
SYSVEC	42#	43					
TABL	540	808#	917	929	1392		
TABLED	92#	789	802	986	1076	1126	1279
TABLES	788	801#	1125				
TABLEX	790	917#					
TBLSEL	484	532#					
TBXFER	183	185#					
TCA	44#	224	227	1086	1195		
TENTHS	901	902	904#				
TICK	596	899#	1386				
TOCK	901#	1061					
UPASCII	126#	568					
UPPER	124#	564					
UPS	515	919#	921	1381			
UPSSZ	516	520	921#				
UPSTRK	481	512#					

UPSX	920#					
USTRK	480	973#				
VER	28#	1155	1155	1155	1394	
WARM	53#	1184				
WD1797	108#	1099	1107			
XNEG	689	975#				
XRBOOT	71#	1323	1326			
XRSIGN	69#	1134	1137			
XY	469	684	757	759	761	926#
YNEG	703	974#				
ZERO	127#	459	729	740		

Notes

A

A66 Control Switch 249, 250
Abbreviations, list of N1
Access cover
- 40 CPS 238, 239
- 20 CPS 227, 228
Access times, disk 207-211
Accessible 1-second interrupt 171, 174, B1
Accessing BIOS 151
Allocation, CRT RAM 45
Alpha lock key
- ASCII 197
- LPK 205, 206
Altitude requirements 18
Application programming 41, 42, 151, 152, 155, 170
ASCII keyboard
ROM, How it Works 133
Dimensions 13
Product Codes 3
General 195-199
CTRL key
Alpha lock key
Auto-repeat keys
Function key priority
Function key uses
Key code chart
Main key array
Numeric key pad
Repeat keystations
Asynchronous, General 135
I/O Port # 157
Port selection 42
Attributes
U57, U58 ROMs 43
Configuration sector 225
Enable, Disable 189
General 183
Set mode 191
Auto-repeat keys
- ASCII 195
- LPK 205

B

BDOS, General 5
BIOS Entry points A2-A5
Banked Physical Driver listing App. I
Device Initialization 133, 134
Execute Physical Driver 161, 162
Op. Sys. Int. 152, 174-176, 178-181
Physical Driver Address Table C3
Programmable Functions 170
Backspace or cursor left 187
Bank switching
Example D4
System PIO 134
Z80-A Example 171, 172
Banked physical driver listing I1-I12
Basic monitor calls D1-D3
Baud rate
I/O Channel 135
Command 144
Generator 136
Host Terminal 145
Port assignments 156
Printer port 135
Table K1

Bell (tone, speaker)
- Control code 187
- Description 47
- Port assignment 157

BIOS
- CP/M-80 2.2 5, 151
- CP/M-86 1.0 151
- Accessing 151, 152
- Cold boot A1
- Console input 167, 168, A2
- Console output 168, A2
- Console status 167, A2
- Entry vector table A1-A5
- Example 152
- Home disk A2
- IOBYTE-directed I/O 167-169
- Interface 151, 152
- Jump table 151
- List device 153, 154, 167-169
- List output A2
- List status A5
- Printer output (IOBYTE) 168
- Printer status (IOBYTE) 169
- Punch output A2
- Read sector A4
- Reader input A2
- Sector translate 152, A5
- Select disk 152, A3
- Set DMA address A4
- Set sector A3
- Set track A3
- Vectors 151
- Warm boot A1
- Write sector A4

Bit mask, keyboard 225

Break switch, printer 240

Bus, system
- Connector 27
- Electrical/environmental 18
- Specification 28-32

Business graphics, (desc.) 16, 17
- CRT control 183
- CRT Controller 43
- CRT entry points 158-160
- CRT font selection 43, 134
- Attributes 44
- Attribute selection 7
- Configuration sector 225
- Display specification 15
- Graphics mode
  - - 40 CPS 243
  - - 20 CPS 232
- Memory allocation 45
- Prog. considerations 186-193
- Scroll register 46

C

CCP, Description 5

CP/M-80, (disk contents) 6-9
- Accessing BIOS 151, 152
- Digital Research files 6, 7
- Entry Vector Table B1
- Prog. considerations A1-A5
- Xerox files 7-9

CP/M-86, (disk contents) 9, 10
- Accessing BIOS 151, 152
- Digital Research files 9, 10
- Xerox files 10

CPU, (general) 2
- ROM 133, 134
- Description 41-44
- Go To Command 140
- System display 183

CRT, (general) 3, 158, 183
- CRT output 158
- RAM 41, 45

Block move 160
Direct CRT display 160
Entry points 158-162
Fast CRT output 159
Font selection 43, 134
IOBYTE 167, 168
Read sector 162
Select media format 161, 162
Set direct CRT cursor 159
Write sector 162
CRT controller, (general) 43, 44
Attributes 16, 17, 44
Control signals (sync & video) 43
Design 16, 17
Font ROMs 43
Functional description 16
CTC (Counter Timer Circuit) 42, 134
Cabling requirements:
- 40 CPS 252
- 20 CPS 236
Capacity, disk 207-211
Carriage return, disp. code 188
Carriage movement
- 40 CPS 242
- 20 CPS 231
Caution, rigid disk 47
Centronics Interface 2, 43, 135
Channel A
SIO Controller 135, 136
Description 42
Entry points 165
Channel B
SIO Controller 135, 136
Description 42
Entry points 163, 164
Character codes, display 186-193
Character delete 192
Character devices 153, 154
Character insert 192
Character sets (information)
ROM 43
Specifications 15, 16
Character sets (pictures)
Graphics 185
Text 184
Character spacing
- 40 CPS 251
- 20 CPS 235
Clear screen 188, 190
Clear to end of line 188, 193
Clear to end of screen 188, 193
Clock
8086 speed 2
1-second interrupt 174
Z80-A speed 2
Code charts K1-K2
Baud rate table K1
Cold boot 151, A1
Cold start 177
Cold start loader listing G1-G6
Command codes:
40 CPS 242-244
Carriage movement
Control codes
Hex codes 245
Margins & formatting
Miscellaneous commands
Paper movement
Printing
Remote Diagnostics
WP commands
20 CPS 231-233
Carriage movement
Control codes
Hex codes 234
Margins & formatting
Miscellaneous commands
Paper movement
Printing
Remote Diagnostics
WP commands
Command set (monitor) 137-149
Summary table 137
Baud rate

Copy memory
Display memory
Extended memory test
Fill memory
Go to
Host Terminal
Input
Load from disk
Modify memory
Output
Protocol
Read disk sector
Typewriter
Verify memory block
Write disk sector

Communications:
SIO Controller 135, 136
SIO Ports, description 42
Baud rate command 144
Comm channel (Port #s) 156, 157
Connector (J4) 19
Host Terminal 144-146
IOBYTE 167, 168
Input Data 165
Input ready status 165
Output status 165
Port (J9), description 5
Protocol command 148, 149
Settings, 40 CPS Printer 249, 250
Strapping Options 25, 26

Configuration sector 225, 226
Configuration status 181
Connectors
(see Hardware Connectors)
Console devices 153, 154
Console input IOBYTE 167
BIOS entry point A2
Console output IOBYTE 168
BIOS entry point A2
Console status IOBYTE 167
BIOS entry point A2

Control codes, display 186-188
Backspace or cursor left
Bell (tone)
Carriage return
Clear to end of line
Clear to end of the screen
Clear screen and home cursor
Cursor down or line feed
Cursor left or backspace
Cursor right
Cursor up
Display next character direct
Escape
Home cursor
Horizontal tab
Line feed or cursor down
Restore previous attribute mode
Set cursor character

Control key, keyboard
- ASCII 197
- LPK 206

Control panel indicators -
40 CPS 239, 240
Break switch
Cover
Form feed
Here is switch
Line feed
Overflow
Parity
Power indicator
Print chk
Printwheel select
Reset
Ribbon/paper
Scroll
Spacing select

Control panel switches -
20 CPS 228
Form feed
Line feed
Pause

Power indicator
Reset
Control sequences, CRT
Codes 187, 188
Table 186
Copy memory,
(Monitor Command) 140
Counter Timer Circuit 134
Cover switch, (40 CPS) 240
CRT memory block move 160
CRT output 158
FAST CRT output 159
CRT RAM
Memory allocation 45
Scroll register 46
CTRL key, keyboard
- ASCII 197
- LPK 206
Cursor, (general) 186
Cursor keys
- ASCII 195
- LPK 200
Cursor control codes
Up, Down, Left, Right 187
Host Terminal cursor control 147

D

DC system power 18
DTR, SIO default 135
Daisy-chain
SIO Controller 42
Rigid/floppy interface 47
Data bits
SIO defaults 135
Baud rate command 144
Configuration sector 225, 226
Get Config Status 181
Host Terminal 144-146
Pass 8 bits/keyboard 190
Pass 7 bits/keyboard 190
Data format, (LPK) 200, 201
Daughter board 47
ROSR monitor
SASI interface
Floppy connectors 38
J1 connector 19
Rigid connector 39
Devices, character & physical
assignments 153, 154
Device names, logical
& physical 153, 154
Diablo printers
- 40 CPS (630) 237-252
- 20 CPS (620) 227-236
Diagnostics, remote printer
- 40 CPS 244
- 20 CPS 233
Digital Research
CP/M-80 files 6-9
CP/M-86 files 9, 10
Dimensions 13
8" Floppy disks
8" Rigid disk
5¼" Floppy disks
40 CPS Printer
20 CPS Printer
ASCII keyboard
Display/Processor
Low profile keyboard
Direct CRT display 160
Disable attributes 189
Disk access connector (J12) 33
Description 34-37
Disk connector (J1) 19
Floppy 38
Illustration 20
Rigid 39
Disk drive format 14

Disk drive storage 14
Disk drives 3, 5
- Access times 207-211
- Caution 47
- Connector 47
- Dimensions 13
- Specifications 207-212
  - Access times
  - Bytes per sector
  - Capacity
  - Encoding method
  - Formats
  - Heat dissipation
  - Latency
  - Power dissipation
  - Recording density
  - Sectors per track
  - Transfer rates
  - Voltage requirements

Disk driver
- Execute physical 161, 162
- Firmware 47
- Media identification 213

Disk mapping
- Description 48
- Device mapping 153, 154
- Drive mapping table C2
- Interleave 223, 224

Disk parameter block (DPB) 214-218
- 8" Floppies
- 8" 10 Mb rigid
- 5¼" Floppies
- Translate tables

Disk parameter header (DPH) 213-218
- Select media format 161
- Value(s) returned 162

Display 3, 15-17, 183-193
- Attributes 16, 183
- Business graphics 16
- Character mode 183
- Dimensions 13
- Graphics character set 185
- Programming considerations 186
- Text character set 184

Display, control codes 186-188
- Backspace or left cursor
- Bell (tone)
- Carriage return
- Clear to end of line
- Clear to end of the screen
- Clear screen and home cursor
- Cursor down or line feed
- Cursor left or backspace
- Cursor right
- Cursor up
- Display next character direct
- Escape
- Home cursor
- Horizontal tab
- Line feed or cursor down
- Restore previous attribute mode
- Set cursor character

Display, functional desc. 16
Display, escape codes 189-193
Display manipulation 186, 189
Display memory command 138
Display next character 188
Display/Processor 13, 183-193
Display specifications 15
- Pixel resolution
- Refresh rate
- Video bit rate
- Video rate

Documented system storage C1, C2
Double density, Select Media Format 161
Downcounter 42
Drive mapping tables C2
Drive selection
- PIO Controller 134
- Description 43

Dual parallel port
PIO Controller 134
Description 43

E

8" Disk access times 207-211
8" Drives, dimensions 13
8" Floppy disks
Double-sided 210, 216
Interleave 223, 224
Rigid 211, 217, 218
Single-sided 209, 215, 216
Track formats 221, 222
820-II Device mapping
General 153, 154
Character devices
IOBYTE
Logical device names
Physical device names
Physical devices
8086
I/O Port assignments 156
CPU Port # 157
Configuration status 181
Description 2, 6, 48
Dual CP/M-80/86 158
Host Terminal 147
Lock instruction 155
Logical/Physical device mapping 154
Memory expansion PWB 132
Operating system files 9-11
Operating system interface 151
Schematics 128-132
System bus connector 27-32
Electrical Requirements
AC 13
820-II-16/8 display
Disk drives
Printers (40 & 20 CPS) 252, 236
DC
Disk drives 207-211
Display 15
Expansion slot 18
(See also Hardware Connectors)
Electrical interface
40 CPS Printer 246-250
A66 control switch
EIA interface
HPR05 circuit board
Dipswitch module A
Dipswitch module B
Enable attributes 189
Enter key, unique codes 204
Entry points, CRT 158-160
Entry vector table A1-A5
Environmental req. 18
Altitude
Humidity
Temperature
Escape codes 189-193
Character delete
Character insert
Clear screen
Clear to end of line
Clear to end of screen
Disable attributes
Enable attributes
Line delete
Line insert
Pass 8 data bits from keyboard
Pass 7 data bits from keyboard
Position the cursor
Set blink attribute
Set graphic mode
Set inverse video
Set low intensity
Escape sequences
CRT 189-193
40 CPS Printer 242, 243
ROSR translations 43

20 CPS Printer 221,222
Etch 1 CPU
J9 - strapping options 25
Schematics 109-116
Serial I/O Ports 42
Etch 2 CPU
J9 - strapping options 26
Keyboard FIFO 48,166
Schematics 117-124
Serial I/O Ports 42
Even parity, host terminal 145
Examples
Bank switching D4
Host Terminal 145,146
List device status 152
Monitor calls/BASIC D1-D3
Protocol command 148,149
Start/Stop 8086 155
Z-80A Assembly Language 171-173
Execute physical driver 161, 162
Expansion slot, (general) 6
Electrical 18
Extended memory
test command 139

F

4.03 ROM 135
40 CPS Printer 237-252
DC1/DC3 237
HPR05
Description 237,238
Interface 247-249
Access cover 238,239
Command codes 242-244
Cont. panel switches 239,240
Electrical interface 246-250
Font selection 249,250
Operating codes 241
Power indicator 239
Specifications 251,252
FIFO 48, 166
Variables C1
Fast CRT output 159
Fill memory command 139
$5\frac{1}{4}$" Disk access times 207, 208
$5\frac{1}{4}$" Drives, dimensions 13
$5\frac{1}{4}$" Floppy disks
Disk Parameter Header 213
Double-sided 208,215
Interleave 223,224
Single-sided 207,214
Track formats 219,220
Floppy disks 213-226
Connector (J1) 19
Illustration 20
Disk parameter block (DPB) 214-218
Disk parameter header (DPH) 213
Interleave 223,224
Track formats 219-222
Font generator, I/O Port # 157
Font ROMs (U57 & U58) 43
Font selection 43, 44, 134
40 CPS Printer 249,250
Form feed
- 40 CPS 239
- 20 CPS 228
Format, floppy disks 219-222
Front cover switches
- 40 CPS 238,239
- 20 CPS 227,228
Function key
Priority - ASCII 196
- LPK 205
Uses - ASCII 196
- LPK 205
Functional description 4, 5
BDOS
BIOS
CCP
ROSR

Operating systems
Ports
System monitor - ROM

G

General Purpose PIO
Ports A & B, Description 43
Specifications 135
Get address of time-of-day variables 180
Get config status 181
Get disk map table address 178, 179
Go to command 140, 141
Graphics character set 185
Mode 183

H

HMI/VMI
- 40 CPS 241
- 20 CPS 230
HPR05 desc. - 40 CPS 237, 238
Hardware connectors 19-39
COMM connector (J4) 19
COMM port (J9) connector
- Etch 1 CPU 25
- Etch 2 CPU 26
Daughter board
- Floppy 38
- Rigid 39
Disk access (J12)
- Description 34-37
- Interface 33
Floppy disk (J1)
- Connector 19
- Illustration 20
- Interface 38
Keyboard connector (J2) 19
Parallel port (J8)
- Connector 23
- Illustration 21
- Picture 24
Power supply (J5 & 6) 22
Printer connector (J3) 19
Rigid disk (J1)
- Connector 19
- Illustration 20
- Interface 39
System bus connector (J13)
- Connector 27
- Description 28-32
Video connectors (J7) 22
Hardware description 2-4
Disk drives
Display
Keyboards
Printers
System board
Hardware handshaking 148
Hardware interface 5, 6
COMM port
Disk drive
Expansion slot
Keyboard
Parallel port
Printer
Hardware scrolling 46
Head Step Rate,
Configur.com 8
Configuration sector 225
Here Is Switch, (40 CPS) 239
Hex codes
- 40 CPS 245
- 20 CPS 234

Home cursor (ESC code) 188
Home disk, BIOS A2
Horizontal Motion Index
- 40 CPS 241
- 20 CPS 230
Horizontal tab - display 187
Host computer 144
Host Terminal mode 144-148
Baud rate
Channel
Command set
Data bits
Monitor output command
Option Summary
Parity
Port
Stop bits
Humidity requirements 18

I

I/O Channel,
setting baud rate 144
IOBYTE
CP/M-80, device mapping 153
CP/M-86, device mapping 154, 155
Console input 167
Console output 168
Console status 167
Directed I/O 167-169
Printer output 168
Printer status 169
Input command (Monitor) 141
Input/output
Port assignments 156, 157
8086
CRT
CTC
GP-PIO channel A
GP-PIO channel B
SIO channel A
SIO channel B
Z80-A
Baud rate
Double density select
Font generator
Scroll register
Single density select
Speaker
Interleave, (floppy disks) 223, 224
Interrupt mode 2 41, C1
Interrupt vectors C1, C2
CTC
GP PIO
SIO
Disk
System PIO
Interrupts
CTC 134
GP PIO 135
SIO 135
Daisy-chain vectoring 42
Disk 34-37
Expansion slot 28-32
System PIO 134

J

J1 (Disk access) 19, 20, 38, 39
J2 (Keyboard) 19
J3 (Printer) 19
J4 (Communications) 19
J5 (Power supply) 22
J6 (Power supply) 22
J7 (Video) 22
J8 (Parallel connector) 23, 24, 43
J9 (Parallel port, Etch 1) 25

J9 (Parallel port, Etch 2) 26
J11 (Option connector) 43, 135
J12 connector 33-37
J13 connector 27-32
Jump vectors 151, 158, A1-A5

K

Key code chart:
- ASCII 198, 199
- LPK 202-204
Keyboards 3, 5, 195-206
FIFO variables C1
ROM 133
ASCII 195-199
Bit mask 224, 225
Connector (J2) 19
Data input 157-165, C1
Device Mapping 153, 154
Dimensions 13
Enter key, unique codes 204
Handler 201
Input (System PIO) 43, 166
Input channel 134
Interface (FIFO variables) 48, C1
Low profile 200-206
Product Codes 3
Status 166
Translation tables 201

L

LST 153, 154, 167-169
(device mapping)
Line delete 192
Line feed switch (40 CPS) 239
Line feed or cursor down 187
Line insert 192
List device 153, 154, 167-169
List output, BIOS A2
List status, BIOS A4
Listings: Appendix
BIOS jump table H
ROM E
Banked physical driver I
Cold start loader G
Macros & Symbols F
Position-encoded kybd handler J
Load from disk command 142
Logical device names
820-II 153
16/8 154
Logical to Physical Disk
Mapping Table 178, 179, C2
Logical/physical
translate tables 213
Low profile keyboard 200-206
Auto-Repeat
Cursor keys
Data format
Dimensions
Enter key, unique codes
Function key uses
Key code chart
Keyboard handler
Main key array
Numeric key pad
Low profile ROM 133, 134, 204

M

MS-DOS 10, 11, 151
Macros & Symbols listing F1-F24

Main key array
- ASCII 195
- LPK 200
Margin & formatting codes
- 40 CPS 242
- 20 CPS 231
Memory allocation 44, 45
Memory banks 41-43
Memory contents 138
Memory pointers C2
Memory size: RAM/ROM 2
Microsoft 10, 11, 151, D1
Miscellaneous functions 177-182
Cold start
Get configuration status
Get disk map table address
Get TOD variable
Start screen print
Warm start
Miscellaneous printer codes:
- 40 CPS 243
- 20 CPS 232
Mode 2 Interrupt Vectors C1, C2
Modem 42, 135, 136
(channel, baud rate generator)
Modes of operation 145
(Host terminal)
Modify memory 138, 139
Monitor calls in BASIC D1-D3
Monitor commands 137-149
Summary table 137
Baud rate
Copy memory
Display memory
Extended memory test
Fill memory
Go to
Host Terminal
Input
Load from disk
Modify memory
Output
Protocol
ROSR
Read disk sector
Typewriter
Verify memory block
Write disk sector
Monitor entry vector table B1
More interrupt vectors C2

N

Numeric key pad
- ASCII 195
- LPK 200

O

1-second interrupt 170, 174
Offset, display 46
Operating codes:
- 40 CPS 241-244
- 20 CPS 230-233
Operating environment 13
Operating mode switches:
- 40 CPS 238-240
- 20 CPS 227-230
Operating System
General 4, 5
Interface 151-182
Output codes, keyboards:
- ASCII 198, 199
- Low Profile 202-204
Output command 141, 142
Output ports 134-136
Output status, comm 165

Overflow Switch (40 CPS) 240

P

PIO, (general) 43, 134, 135
Paper movement, printer:
- 40 CPS 243
- 20 CPS 231
Parallel I/O controller 134, 135
  Bank switching
  Floppy drive select
  Floppy side select
  Font selection
  Keyboard input channel
Parallel port 2, 6, 43
  Connector 23, 24
Parallel printer 43, 135
Parity
  Host Terminal 145, 146
  40 CPS switch 240
  20 CPS switch 229
Pass upper bit of data from keyboard 190
Pause switch (20 CPS) 228
Physical device names
  820-II 153
  16/8 154
Physical Disk Interleave 223, 224
Physical driver
  Address table C3
  Request block 161, 162, 176, C3
Physical format, disk drives 219-222
Physical to Logical Device Mapping 167
Pixel resolution 15
Ports
  Communication 4, 5, 19, 25, 26, 135
  General 4
  Parallel 23, 24
  Printer 4, 5, 19, 135
Position-encoded keyboard
  Handler 201
  Listing J1-J38
Position the cursor 191, 192
Power indicator:
- 40 CPS 239
- 20 CPS 228
Power on sequence 133-136
Power supply (J5 & 6) 22
Print Chk switch (40 CPS) 240
Printers, general 4, 227-252
  A66 Control switch 249, 250
  40 CPS 237
  20 CPS 227
  Baud rate command 144
  Baud rate generator 136
  Buffer 238, 251
  Channel 42
  Connector (J3) 19
  Dimensions 13
  Entry points 163, 164
  Output 163, 164
  Overview 163
  Port 42, 43, 135, 136
  Status 163
Printing commands
- 40 CPS 243
- 20 CPS 232
Printwheel selection (40 CPS) 238
Product code numbers 3, 4
Programmable functions: 170-176
  1-second interrupt
  Processing while I/O pending
  Sample program
  Soft disk error
  Soft error recording
  System exit points
Programming considerations:
  BIOS A1-A5
  CRT Control codes 187, 188

CRT Escape codes 189-193
Display 186-193
Escape sequences
- 40 CPS 242, 243
- 20 CPS 231, 232
Printer control codes
- 40 CPS 245
- 20 CPS 234
Proportional spacing (HMI/VMI)
- 40 CPS 241
- 20 CPS 230
Protocol command 148, 149
XON/XOFF
Hardware handshake
Serial printers
Punch output, BIOS A2

R

RAM, CRT 41-46
Attributes
Driver
Font Selection
Memory Allocation
Offset
Power on
Refresh RAM
Scroll Register
ROM 5, 41, 42, 133-136, 158
ROM listings E1-E110
ROM/O.S. Interface 158
ROSR 5, 41, 44, 47, 133-136, 170
RS232 (See Comm port)
RTS 135, 136
Read disk sector command 143
Read sector 162, A5
Reader input A2
Real Time Clock 174
(See also Clock)
Refresh RAM 43
Memory 183
Rate 15
Remote Diagnostics:
- 40 CPS 243, 244
- 20 CPS 232, 233
Repeat keystations:
- ASCII 195
- LPK 205
Reserved
ASCII Keys 198, 199
Low Profile Keys 202-204
Ports 157
Tracks 214-218
Reset Switch (40 CPS) 239
Reset, Power on 133-136
Resident monitor commands 137-149
Resolution, screen 15
Restore previous attribute mode 187
Ribbon/Paper switch (40 CPS) 240
Rigid disk conn. (J1) 19, 20, 39

S

7 data bits (SIO controller) 110
1797 - Western Digital 279-302
6-8k ROM 2, 42
16/8 physical device mapping 153, 154, 167-169
Application programs
Communication channel
IOBYTE
Logical device names
Physical device names
Physical devices
64k RAM 2, 42
ROSR monitor

Z80-A CPU
SA 1403
Processing while I/O pending 174
Soft Error Recording 176
SA 1403D controller 254-278
SA400L 207, 214, 219, 220, 223, 224
SA450 208, 215, 219, 220, 223, 224
SA800 209, 215, 216, 221, 224
SA850 210, 216, 221, 224
SA1004 211, 217, 218, 223, 224
SASI interface 47
SIO, (general) 42
SIO Channels:
- A 135, 136
- B 135, 136, 163, 164
SIO controller 42, 135, 136
SIO-B
Input data 163
Input ready status 163
Output data 163
Output ready status 164
Sample programs
Z80-A assembly 171-173
Monitor calls - BASIC D1-D3
Bank switching D4
Schematics 109-132
Screen attributes - see Attributes
Screen codes 186-193
Screen print 182
Screen resolution:
Description 16
Specification 15
Scroll switch (40 CPS) 239
Scroll register 45, 46
Hardware scrolling
Offset
Sectors per track 207-211
Sector translate, BIOS A5
Table 213
Vector 152
Sectran 152, A5
Seldsk 152, A3
Select media format 161
Serial I/O Controller 42, 135, 136
Async modem port
Baud rate
Channel B
Printer port
Set blink attribute 191
Set cursor character 187
Set DMA address A4
Set direct CRT cursor 159
Set graphic character 191
Set inverse video 191
Set low intensity 191
Set sector A3
Set track A3
Shugart - 1403D 253-278
Side selection 43, 134, 161
Single density 161
Skewing 223, 224
Soft disk error 176
Soft error recording 176
Spacing select (40 CPS) 238
Speaker (bell, tone) 47, 157
Specifications
- 820-II & 16/8 13-16
Dimensions
Disk drive format
Disk drive storage
Display
Electrical
Operating environment
Specifications
40 CPS 251, 252
Cabling requirements
Character set
Character spacing
Power requirements
Print buffer
Print line
Printwheels
20 CPS 235, 236
Cabling requirements

Character set
Character spacing
Power requirements
Print buffer
Print line
Printwheels
Start screen print 182
Storage capacity 207-211
Strapping options,
Comm port 25, 26
Strobe, printer 135
Synchronous
I/O Port # 157
Port selection 42
System board 2
System bus 18, 48
Connector (J13) 27-32
System configuration 181
System display 183-193
Graphics character set
Programming considerations
Text character set
System exit points 170-176
System monitor,
ROM 4, 133-136
System PIO 43, 134
System storage C1

T

20 CPS Printer
Dimensions 13
General 227-237
Cabling requirements
Command codes
Control panel switches
Front cover switches
Horizontal motion index (HMI)
Power indicator
Operating codes
Operating mode switches
Specifications
Vertical motion index (VMI)
Temperature req. 18
Text character set 184
Theory of Operation 41-48
6-8k ROM memory
64k RAM
CPU
CRT controller
CTC
PIO Controller
SIO Controller
Bus expansion
Daughter board
Dual parallel ports
Memory allocation
Parallel keyboard interface
Scroll register
Serial I/O ports
Speaker
Time constant
Time-of-day and
timer variables 47, 180, C4
Timer variables 180, C4
Tone generator 47, 157
(speaker, bell)
Track format, disk
drives 207-211
Transfer rates 207-211
Typewriter command 144

V

Vector table B1
Verify command 140
Vertical Motion Index:
- 40 CPS 241
- 20 CPS 230

Video bit rate 15
Video connector (J7) 22
Video rate 15

W

WD1797 174, 176, 279-302
Warm boot A1
Warm start 177
Western Digital
- General 174, 176
- Reprint 279-302

WP commands:
- 40 CPS printer 243
- 20 CPS printer 232

Write disk sector command 143
Write sector, BIOS 162, A4

X

XON/XOFF protocol 148
Xerox files
- CP/M-80 disk 7-9
- CP/M-86 disk 10

Xtended memory test command 139

Z

Z80-A 2, 41, 42, 47-108 133, 158, C1, C2
- Assembly language ex. 171-173

Zilog Reprint 49-108